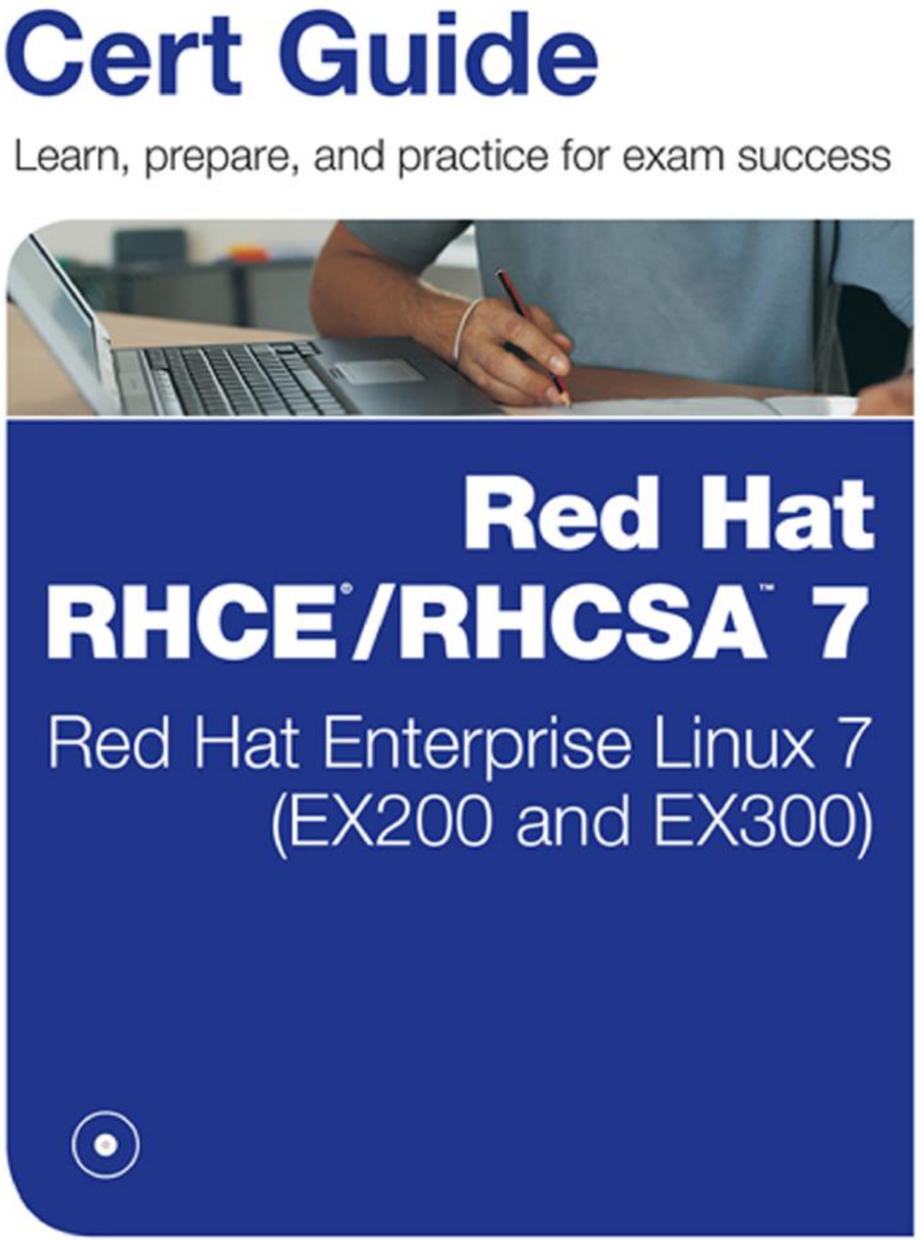
RHCSA - EX200


Cert Guide
Learn, prepare, and practice for exam success
Red Hat
RHCE®/RHCSA™ 7
Red Hat Enterprise Linux 7
(EX200 and EX300)

Course Materials
Written by
SANDER VAN VUGT

Trainer

Ali Aydemir

CISCO, CCIE #47287 SP/RS, CCSI#35413
AVAYA, ACE-Fx #169
HUAWEI, HCDP

RHCSA Timetable

Day	AM	Lunch	PM
1	-Installing RHEL Server -Using Essential Tools	--	-Essential File Management -ToolsWorking with Text Files -Connecting to a RHEL Server
2	-User and Group Management -Permissions Management	--	-Configuring Networking -Process Management -Working with Virtual Machines
3	-Installing Software Packages -Scheduling Tasks	--	-Configuring Logging -Managing Partitions -Managing LVM Logical Volumes
4	-Basic Kernel Management -Configuring a Basic Apache Server	--	-Managing and Understanding the Boot Procedure -Essential Boot Procedure Troubleshooting
5	-Managing SELinux -Configuring a Firewall	--	-Configuring Remote Mounts and FTP -Configuring Time Services

Chapter 1:

Installing RHEL Server

Chapter 1 Objectives

- The following topics are covered in this chapter:
  - Preparing to Install Red Hat Enterprise Linux
  - Performing a Manual Installation

- This chapter covers no exam objectives.

What Is Red Hat Enterprise Linux 7 Server?

- RHEL 7 is a Linux distribution. As you probably know, Linux is a free operatingsystem. That means that the source code of all programs is available for free, which is also the case for Red Hat Enterprise Linux 7. However, you cannot download RHEL 7 for free (with the exception of a 30-day evaluation version).

- To use RHEL 7, you need a subscription. This subscription entitles you to a few additional items, such as support and patches. When you pay for Red Hat Enterprise Linux, Red Hat offers you a supported Enterprise Linux operating system.

Using CentOS

- CentOS 7 is the Community Enterprise Operating System. CentOS 7 started as a recompiled version of RHEL, where all items that were not available for free were removed from the software. Basically, that meant that just the name was changed and that the Red Hat logo (which is proprietary) was removed from all the CentOS software packages. The result was an operating system that offered exactly the same functionality as RHEL 7 but was available for free (and without the enterprise support services).

- Another benefit of CentOS is that it has a bigger hardware compatibility list than RHEL 7, because more drivers are compiled in the CentOS 7 kernel.

Understanding Access to Repositories

- An important difference between RHEL and the other distributions is the access to repositories. A repository is the installation source used for installing software. If you are using free software such as CentOS, correct repositories are automatically set up, and no further action is required.

Setup Requirements

- 64-bit platform support
- 1GB of RAM
- A 10GB hard disk
- A DVD drive
- A network card

Course Environment Description

- To perform all exercises in this course, recommend installing three different servers, named as follows:
  - Server1
  - Server2
  - FreeIPA

- Although recommend setting up all parts that are required for this course manually, if you are having a problem you might like the environment that provide at http://www.rhatcert.com/downloads/ Go to the download area of this website to get access to all software required to work on the exercises in this book.

Course Environment Description

- Server1 is a base RHEL 7 installation used in the RHCSA part of this course.
- Server2is used when client/server functionality needs to be tested.
- The FreeIPA server isrequired in some of the labs related to authentication and Network File System(NFS).
- To install server1 and server2, you can follow the generic guidelines in thischapter; all specific parts of the setup are explained in the specific exercises. To install the FreeIPA server, you can follow the directions in “Setting UpIdentity Management,” after you have installed a base installation of RHEL 7.

Performing a Manual Installation

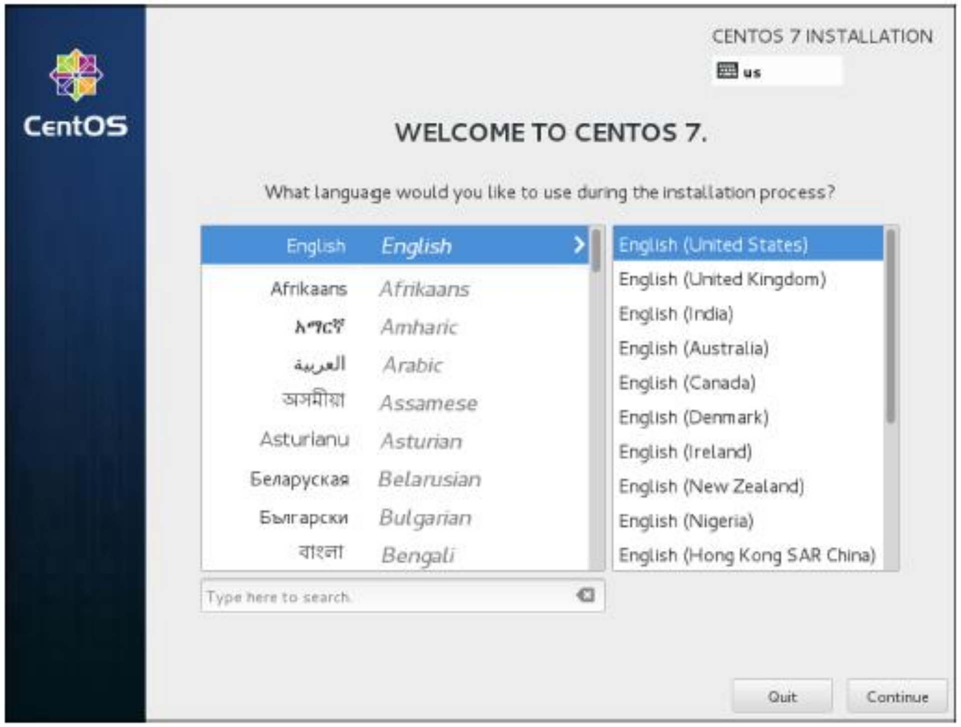
1. After booting from DVD, you'll see the CentOS 7 boot menu. From this menu, you can choose from different options:

- ❑ **Install CentOS 7**: Choose this for a normal installation.
- ❑ **Test This Media & Install CentOS 7**: Select this if before installing you want to test the installation media. Notice that this will take a significant amount of time.
- ❑ **Troubleshooting**: Select this option for some troubleshooting options. This option is useful if you cannot normally boot from your computer's hard drive anymore.

To start a normal installation, just select the **Install CentOS 7** boot option.

Performing a Manual Installation

2. Once the base system from which you will perform the installation has loaded,you see the Welcome to CentOS 7 screen. From this screen, you can select the language and the keyboard setting.



WELCOME TO CENTOS 7.

What language would you like to use during the installation process?

English	English
Afrikaans	Afrikaans
አማርኛ	Amharic
العربية	Arabic
অসমীয়া	Assamese
Asturianu	Asturian
Беларуская	Belarusian
Български	Bulgarian
বাংলা	Bengali

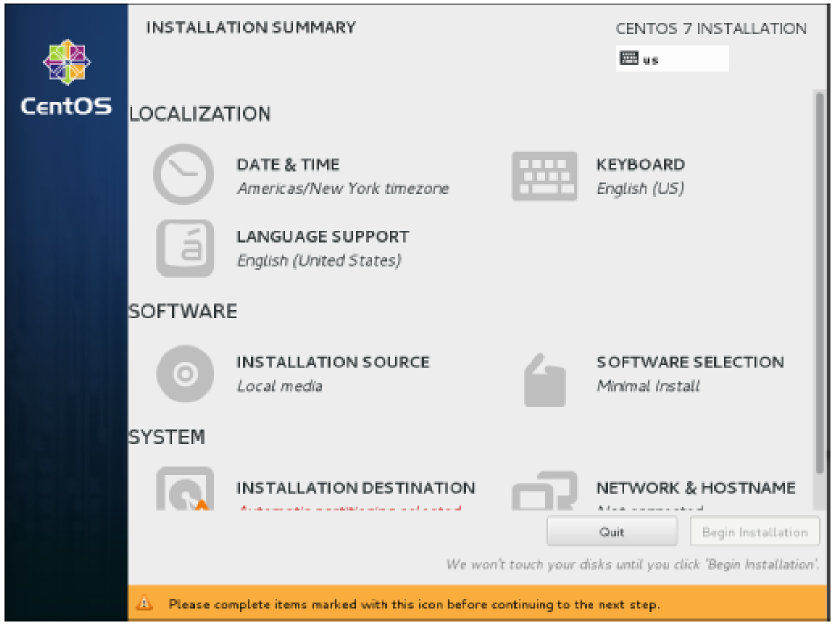
English (United States)
English (United Kingdom)
English (India)
English (Australia)
English (Canada)
English (Denmark)
English (Ireland)
English (New Zealand)
English (Nigeria)
English (Hong Kong SAR China)

Figure 1.1 Select the appropriate keyboard setting before continuing.

Performing a Manual Installation

3. After selecting the keyboard and language settings, you'll see the Installation Summary screen (see Figure 1.2). From this screen, you specify all settings you want to use. On this screen, you have seven different options:

- Date & Time
- Keyboard
- Language Support
- Installation Source
- Software Selection
- Installation Destination
- Network & Hostname



DATE & TIME
Americas/New York timezone

KEYBOARD
English (US)

LANGUAGE SUPPORT
English (United States)

INSTALLATION SOURCE
Local media

SOFTWARE SELECTION
Minimal Install

INSTALLATION DESTINATION

NETWORK & HOSTNAME

We won't touch your disks until you click 'Begin Installation'.

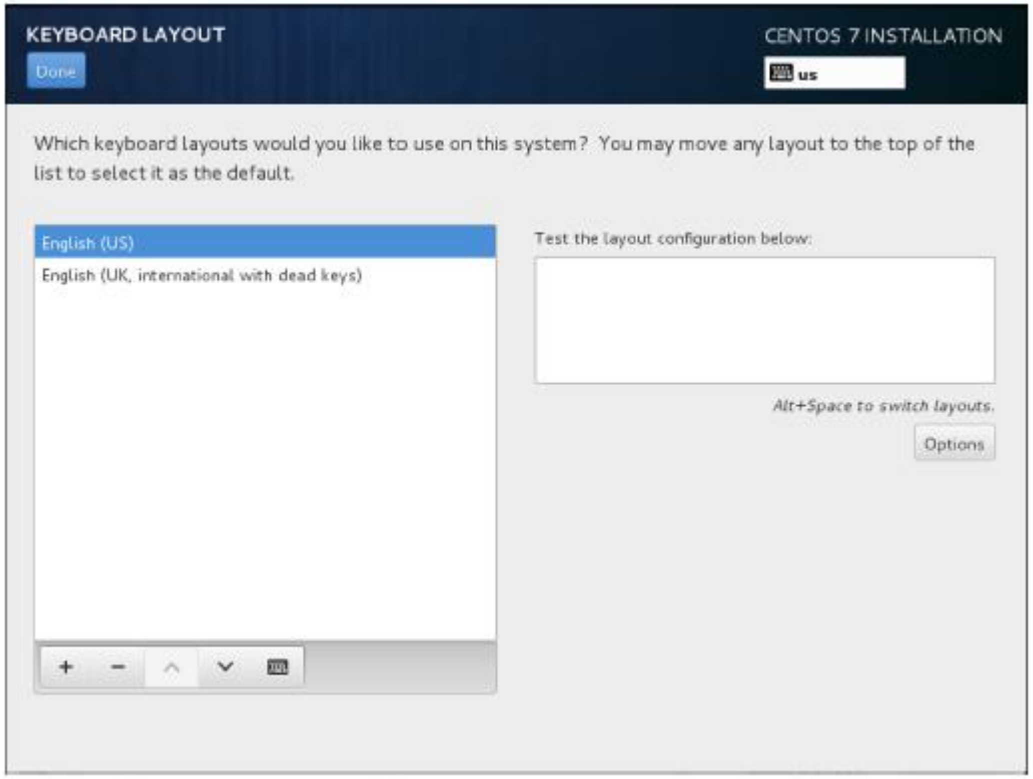
Please complete items marked with this icon before continuing to the next step.

Figure 1.2 You specify the complete configuration of your server from the Installation Summary screen.

Performing a Manual Installation

4. After selecting Date & Time, you’ll see a map of the world on which you can easily click the time zone that you are in (see Figure 1.3). Alternatively, you can select the region and city you are in. You can also set the current date and time, and after setting the network, you can specify the Network Time Protocol (NTP) to be used. This option is not accessible if you have not configured the network yet. When using network time, you can add network time servers to be used by clicking the configuration icon in the upper-right part of the screen. After specifying the settings you want to use, click Done in the upperleft corner of the screen to write the settings.

Performing a Manual Installation



Which keyboard layouts would you like to use on this system? You may move any layout to the top of the list to select it as the default.

English (US)
English (UK, international with dead keys)

Test the layout configuration below:

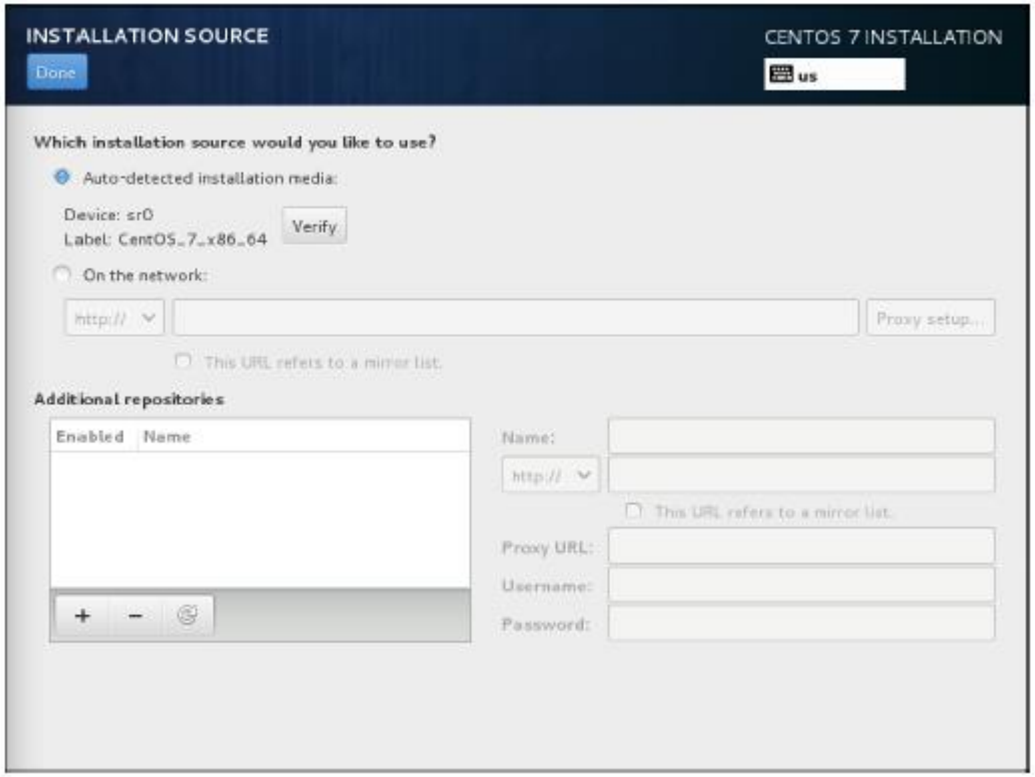
Alt+Space to switch layouts.

Figure 1.3 Selecting date and time settings.

Performing a Manual Installation

5. Under the Keyboard Layout option, you'll find what you need to configure the keyboard layout. From this screen, shown in Figure 1.4 , you can also select a secondary keyboard layout, which is useful if your server is used by administrators using different keyboard layouts. Not only different language settings are supported, but also different hardware layouts. If many administrators are using an Apple Macintosh computer, for instance, you can select the standard keyboard layout for Mac in the appropriate region.
6. The Language Support option is the same as the Language Support option that you used in Step 2 of this procedure. If you've already configured the language settings to be used, you do not need to change anything here.

Performing a Manual Installation



Which installation source would you like to use?

● Auto-detected installation media:

Device: sr0
Label: CentOS_7_x86_64

○ On the network:

☐ This URL refers to a mirror list.

Additional repositories

Enabled	Name

Name:

☐ This URL refers to a mirror list.

Proxy URL:

Username:

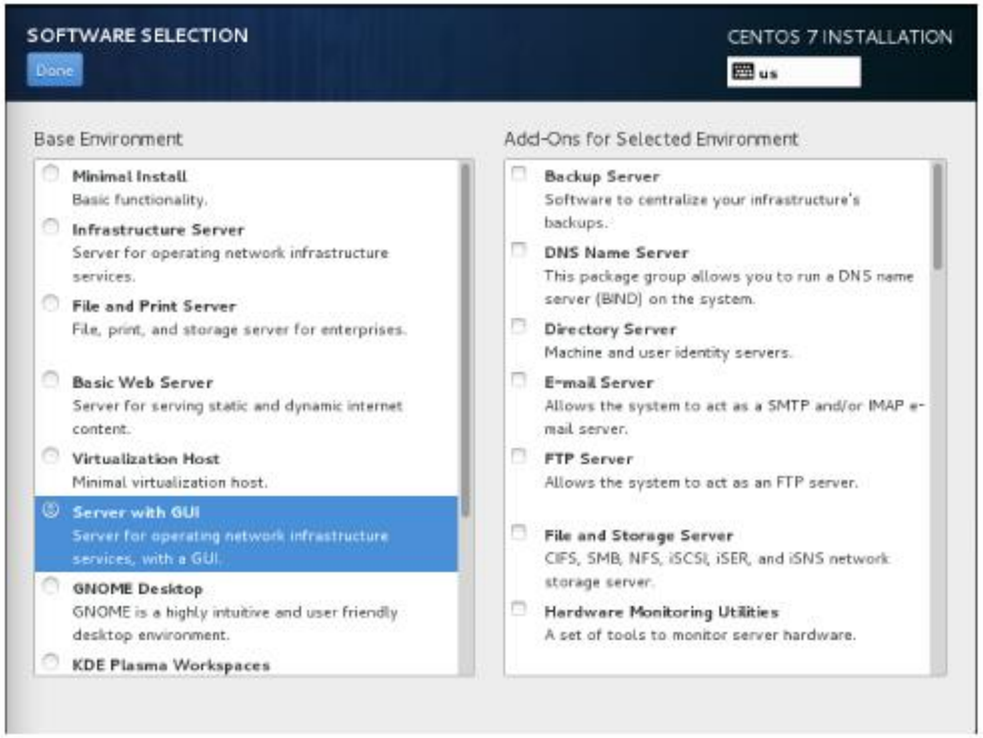
Password:

Figure 1.4 Selecting additional keyboard layout.

Performing a Manual Installation

7. In the Software section, the first option available is Installation Source (see Figure 1.5). If you have booted from a regular installation disc, there is nothing to be specified from this option. If you have booted from a minimal boot environment, you can specify the network URL where additional packages are available, as well as additional repositories that need to be used. You do not have to do this for the RHCSA or RHCE exam, but if ever you are setting up an installation server, it is useful to know that this option exists.

Performing a Manual Installation


SOFTWARE SELECTION

CENTOS 7 INSTALLATION

Base Environment

- **Minimal Install** Basic functionality.
- **Infrastructure Server** Server for operating network infrastructure services.
- **File and Print Server** File, print, and storage server for enterprises.
- **Basic Web Server** Server for serving static and dynamic internet content.
- **Virtualization Host** Minimal virtualization host.
- [✓ **Server with GUI**] Server for operating network infrastructure services, with a GUI.
- **GNOME Desktop** GNOME is a highly intuitive and user friendly desktop environment.
- **KDE Plasma Workspaces**

Add-Ons for Selected Environment

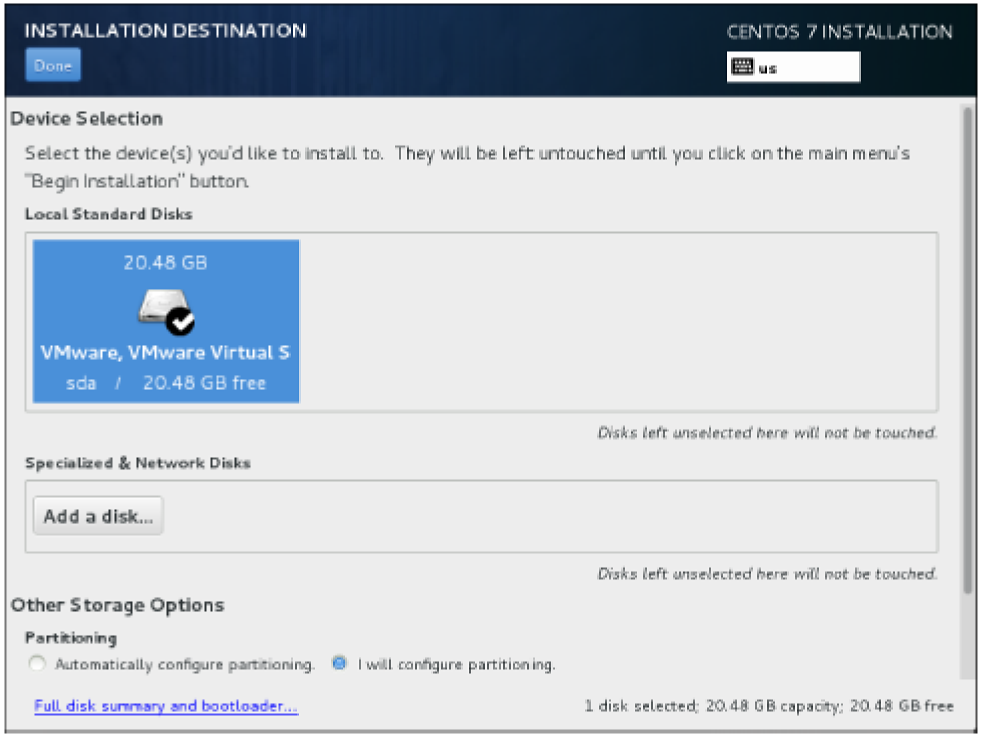
- **Backup Server** Software to centralize your infrastructure's backups.
- **DNS Name Server** This package group allows you to run a DNS name server (BIND) on the system.
- **Directory Server** Machine and user identity servers.
- **E-mail Server** Allows the system to act as a SMTP and/or IMAP e-mail server.
- **FTP Server** Allows the system to act as an FTP server.
- **File and Storage Server** CIFS, SMB, NFS, iSCSI, iSER, and iSNS network storage server.
- **Hardware Monitoring Utilities** A set of tools to monitor server hardware.

Figure 1.5 Selecting the installation source.

Performing a Manual Installation

8. An important part of the installation procedure is the Software Selection screen. From here, you select the base environment (see Figure 1.6) and addons that are available for the selected environment. By default, a Minimal Installation is selected. This base environment allows you to install RHEL on a minimal-size hard disk. For this course, I assume that you install the Server with GUI option. To perform the tasks that need to be performed on the RHCSA and RHCE exams, some easy-to-use graphical tools are available, so it does make sense to install a server with a graphical user interface (GUI) (even if you would never do this in a production environment). All additional packages can be added later. At this point, you do not have to select any additional packages.

Performing a Manual Installation



Device Selection

Select the device(s) you'd like to install to. They will be left untouched until you click on the main menu's "Begin Installation" button.

Local Standard Disks

20.48 GB
VMware, VMware Virtual S
sda / 20.48 GB free

Disks left unselected here will not be touched.

Specialized & Network Disks

Add a disk...

Disks left unselected here will not be touched.

Other Storage Options

Partitioning

Automatically configure partitioning. [✓ I will configure partitioning.]

Full disk summary and bootloader...

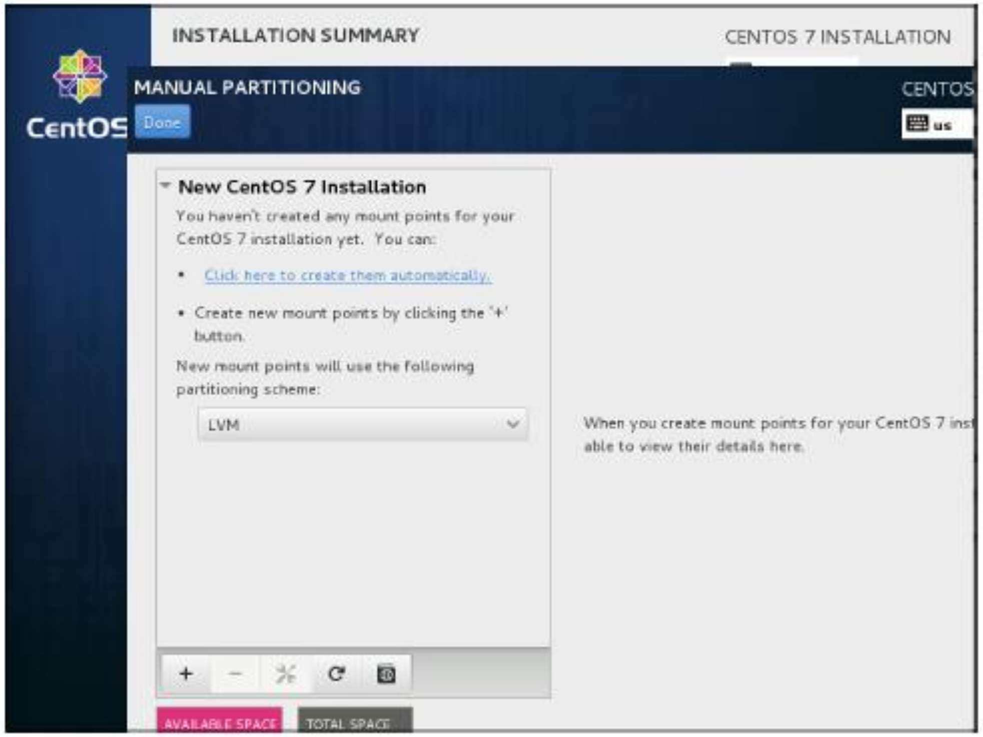
1 disk selected; 20.48 GB capacity; 20.48 GB free

Figure 1.6 Make sure you select **Server with GUI** for every server you are going to use for the exercises in this book.

Performing a Manual Installation

9. Next, you need to specify where you want to install to. By default, automatic partitioning is selected and you only need to approve the disk device you want to use for automatic partitioning (see Figure 1.7). Many advanced options are available, as well. To prepare your installation for all the exercises that are in later chapters in this book, you cannot just use the default partitioning. Instead, you need a setup that uses LVM and also keeps some disk space available. To do this, from the screen you see in Figure 1.7 , select I Will Configure Partitioning . Then, make sure that the disk you want to use is selected and click Done to proceed. Notice that sometimes not all options are shown, and you'll see a scrollbar to the right of the screen. If this is the case, scroll down to show additional installation options. You'll see the Encryption option, which allows you to set up an encrypted disk.

Performing a Manual Installation



New CentOS 7 Installation

You haven't created any mount points for your CentOS 7 installation yet. You can:

- Click here to create them automatically.
- Create new mount points by clicking the '+' button.

New mount points will use the following partitioning scheme:

LVM

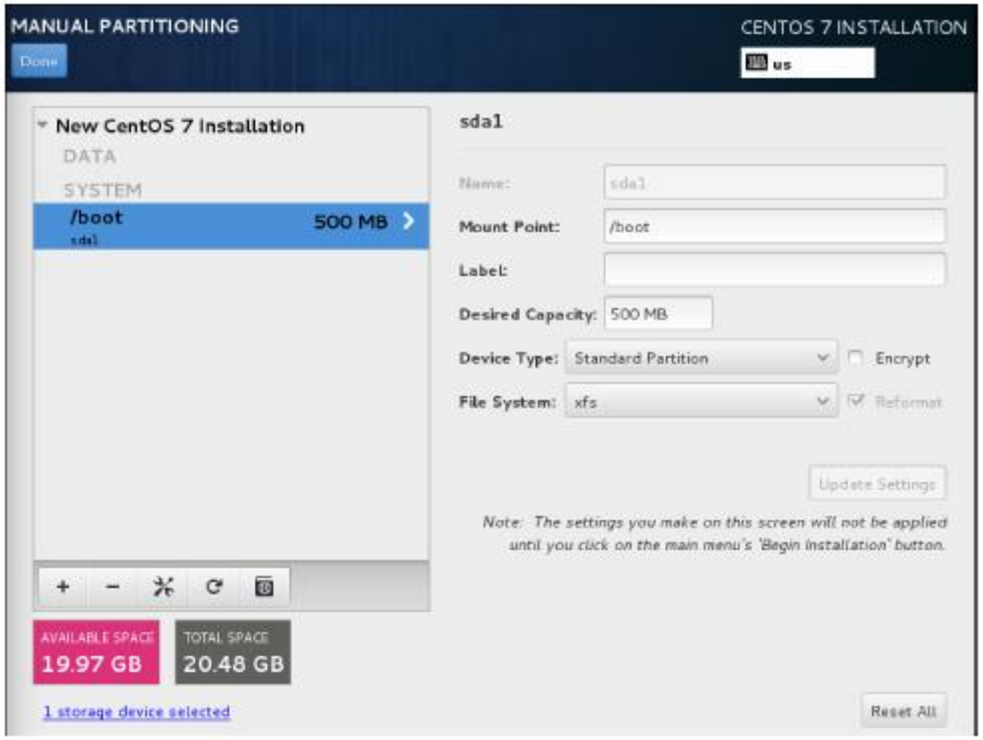
When you create mount points for your CentOS 7 ins
able to view their details here.

Figure 1.7 Select **I Will Configure Partitioning**, and click **Done** to proceed.

Performing a Manual Installation

- After specifying that you want to set up disk layout manually, you’ll see the screen that is in Figure 1.8 . From this screen, click **+** to add new disk devices.
- For setting up the environment that is required in this course, recommend using the following disk layout (based on a 20GB hard disk).
  - /boot mounted on an XFS formatted traditional partition, size 500MB
  - An XFS formatted logical volume with a size of 10GB that is mounted on /
  - A 1GB logical volume that is used as swap space

Performing a Manual Installation



sda1

Name:	sda1
Mount Point:	/boot
Label:	
Desired Capacity:	500 MB
Device Type:	Standard Partition ☐ Encrypt
File System:	xfs ☑ Reformat

Note: The settings you make on this screen will not be applied until you click on the main menu's 'Begin Installation' button.

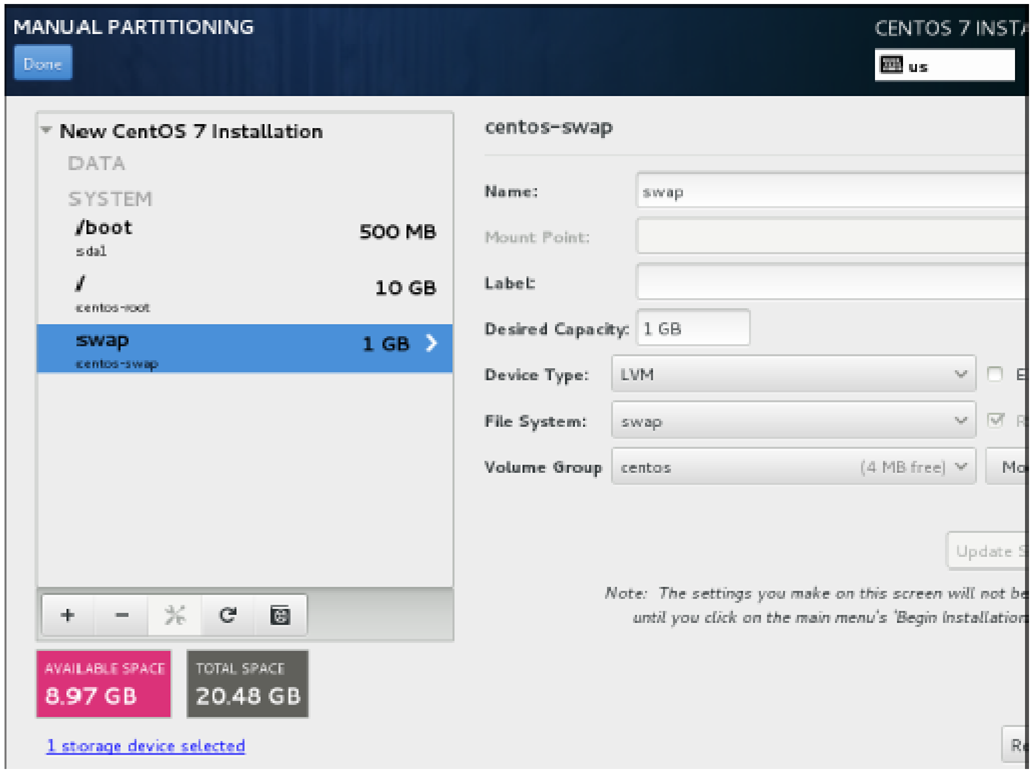
Figure 1.8 Configuring advanced disk layout.

RHEL 7 by default uses the **XFS** file system. This file system cannot be *shrunk*; it can only be expanded. Therefore, it is sometimes a better choice to use **ext4**.

Performing a Manual Installation

- To create this configuration, from the screen in Figure 1.8 , click the + sign. You’ll now see a pop-up in which you can specify a mount point and the desired capacity. From the Mount Point drop-down list, select /boot and add the desired capacity 500M . Then, click Add Mount Point . You’ll now get to the screen that you see in Figure 1.9 , in which you can specify specific details about the mount point you just created.

Performing a Manual Installation



New CentOS 7 Installation
DATA
SYSTEM
/boot sda1 500 MB
/ centos-root 10 GB
swap centos-swap 1 GB

AVAILABLE SPACE 8.97 GB
TOTAL SPACE 20.48 GB

centos-swap

Name: swap
Mount Point:
Label:
Desired Capacity: 1 GB
Device Type: LVM
File System: swap
Volume Group centos (4 MB free)

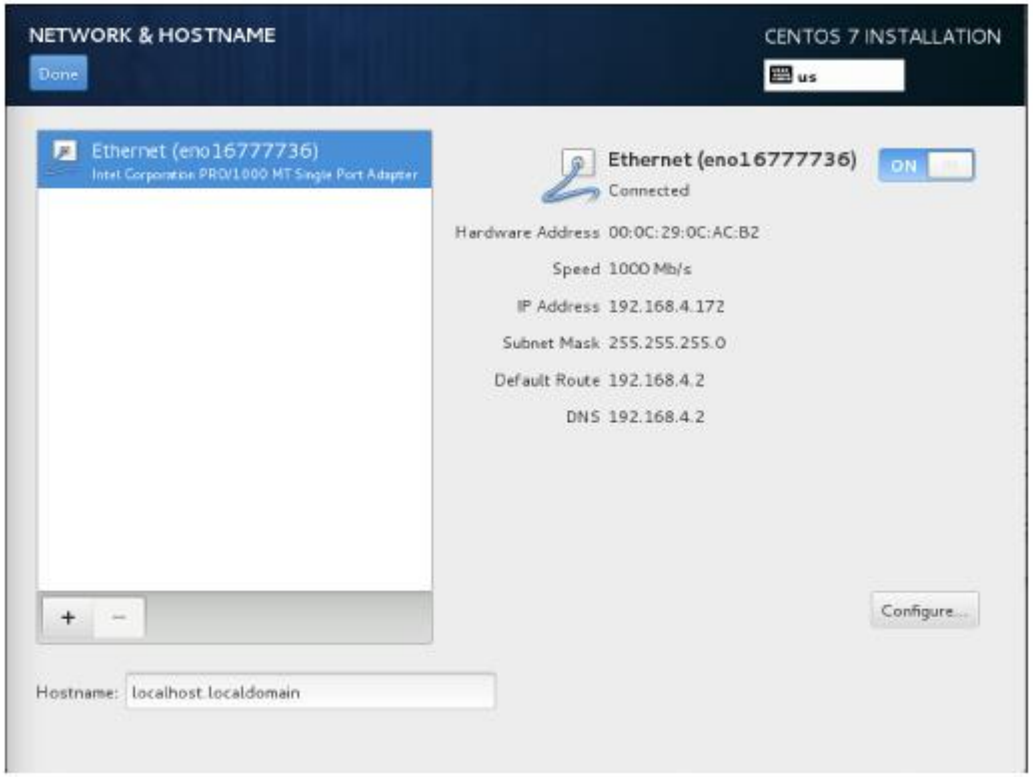
Note: The settings you make on this screen will not be
until you click on the main menu's 'Begin Installation

Figure 1.9 Specifying details for the mount point you just created.

Performing a Manual Installation

- At this point, from the interface that you see in Figure 1.9 , click the + sign again and specify the mount point / and a capacity of 10G . On the Mount Point Details screen that you see now, make sure that the device type LVM is set and that the file system XFS is selected. You do not have to modify anything else.
- Now click the + sign once more to add a swap device. From the Mount Point drop-down list, select swap , and specify the desired capacity of 1G . Then, click Add Mount Point . The layout at this point should look like Figure 1.10 . If this is the case, click Done to write the configuration.

Performing a Manual Installation



Ethernet (eno16777736)
Connected

Hardware Address 00:0C:29:0C:AC:B2
Speed 1000 Mb/s
IP Address 192.168.4.172
Subnet Mask 255.255.255.0
Default Route 192.168.4.2
DNS 192.168.4.2

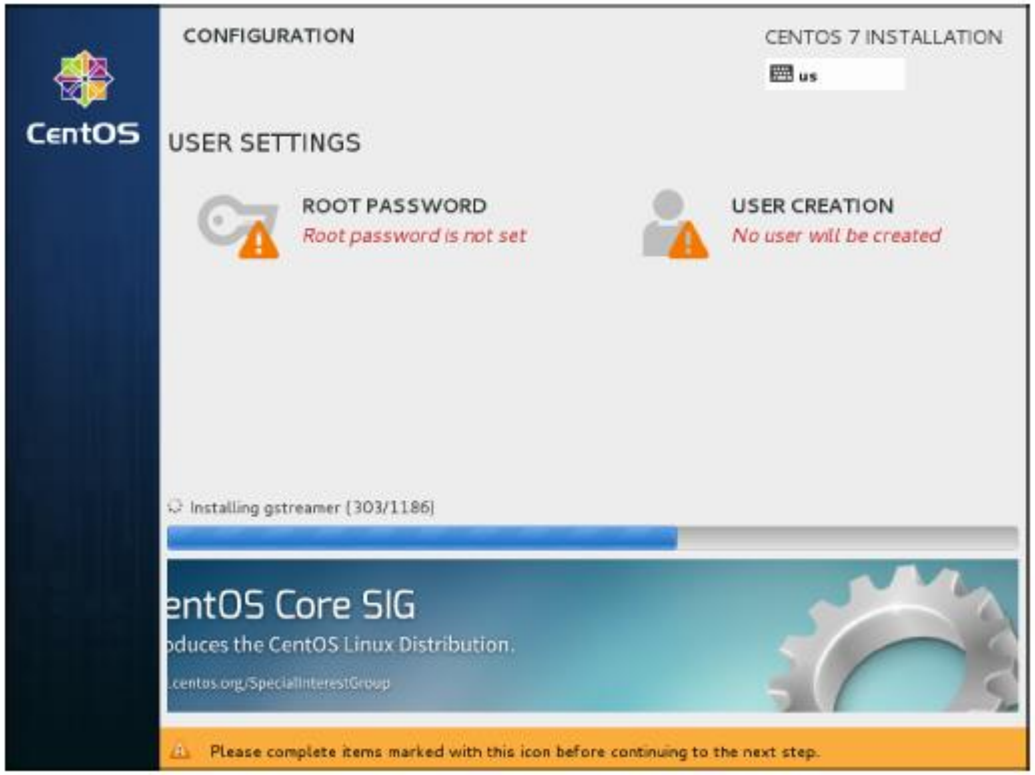
Hostname: localhost.localdomain

Figure 1.10 If the configuration looks like this, you can write it to disk.

Performing a Manual Installation

10. The last part of the installation summary allows you to set up networking. Notice that you must configure something. If you do not do anything, your server will not be able to connect to any network. From the Installation Summary screen, click Network & Hostname to set up networking. This shows the screen that you see in Figure 1.11 .

Performing a Manual Installation


CONFIGURATION

CENTOS 7 INSTALLATION

us

USER SETTINGS

ROOT PASSWORD
Root password is not set

USER CREATION
No user will be created

Installing gstreamer (303/1186)

entOS Core SIG
oduces the CentOS Linux Distribution.
.centos.org/SpecialInterestGroup

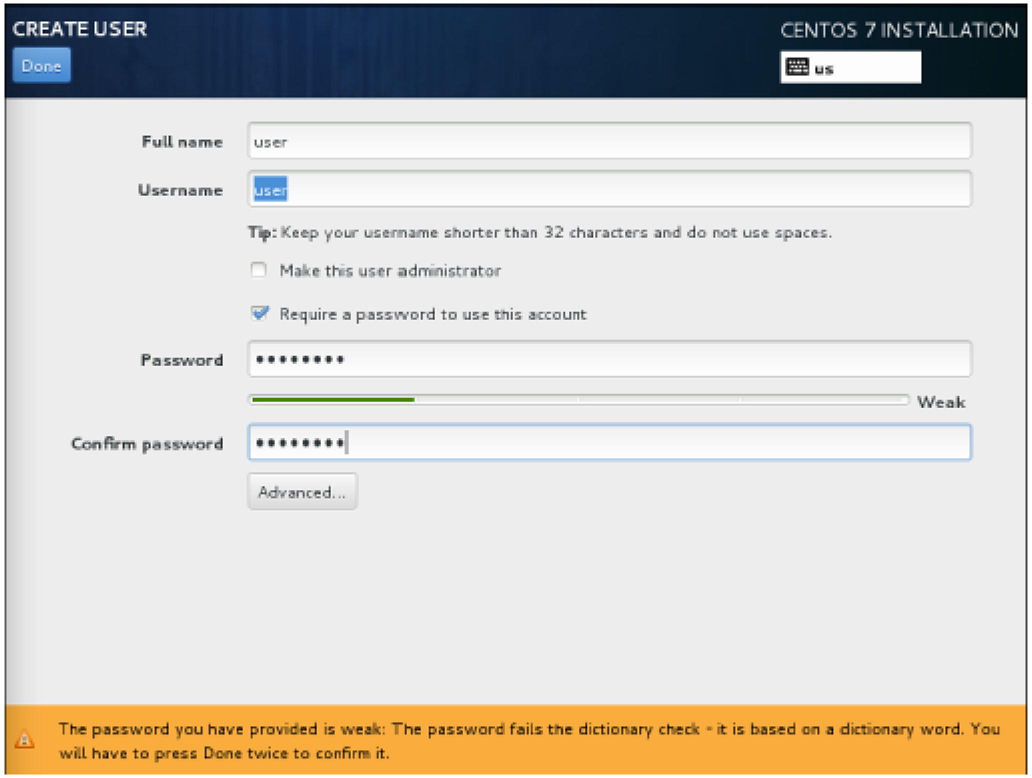
Please complete items marked with this icon before continuing to the next step.

Figure 1.11 In the Network & Hostname screen, you must set the network card to on.

Performing a Manual Installation

11. After specifying all settings in the installation summary, you can click **Begin Installation** to start the installation. This immediately starts the installation procedure but also prompts for user settings (see Figure 1.12). From this screen, click **Root Password** first, and set the password to **password**. That is not very secure, but by using a simple password like this, you'll avoid the issues later on that might result from not remembering the password. You have to specify the password twice, and you also need to click **Done** twice. This is because you have to confirm that you really want to use a weak password. Next, click **User Settings** to create a user. Enter the full name and username you want to use, and for this user, set the password to **password** also. Again, you have to click **Done** twice to confirm that you really want to use a weak password.

Performing a Manual Installation



Full name: user

Username: user

Tip: Keep your username shorter than 32 characters and do not use spaces.

☐ Make this user administrator

✓ Require a password to use this account

Password: ••••••••

Weak

Confirm password: ••••••••

The password you have provided is weak: The password fails the dictionary check - it is based on a dictionary word. You will have to press Done twice to confirm it.

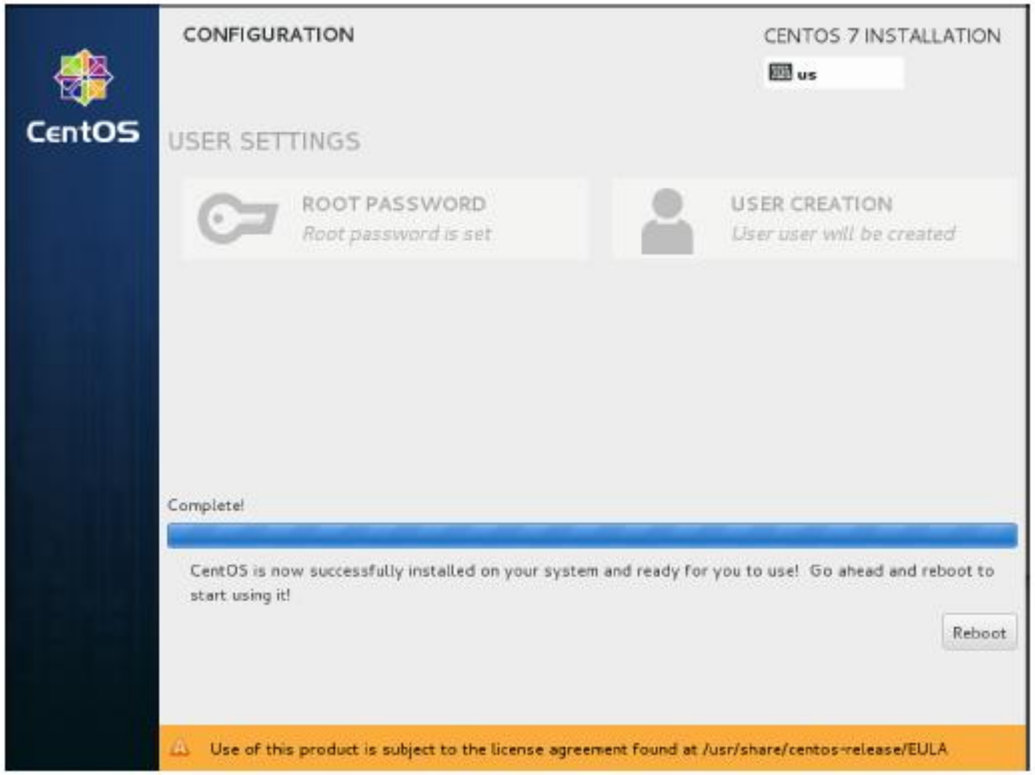
Figure 1.12 Specifying additional user settings.

Performing a Manual Installation

12. When the installation has completed, you'll see the screen shown in Figure 1.13 . You'll now need to click Reboot to restart the computer and finalize the installation.

13. After rebooting, you have to go through a couple of additional setup steps. Fist, you need to accept the license agreement. To do this, click the red text License Not Accepted , select I Accept the License Agreement , and then click Done to complete. You can now click Finish the Configuration to finalize the configuration, which brings you to the graphical login prompt.

Performing a Manual Installation



Complete!

CentOS is now successfully installed on your system and ready for you to use! Go ahead and reboot to start using it!

Use of this product is subject to the license agreement found at /usr/share/centos-release/EULA

Figure 1.13 Reboot to finalize the installation.

Summary

- In this chapter, you learned what Red Hat Enterprise Linux is and how it relates to some other Linux distributions.
- You also learned how to install Red Hat Enterprise Linux 7.
- You are now ready to set up a basic environment that you can use to work on all the exercises in this course

Define Key Terms

- Define the following key terms:
  - Distribution
  - Linux
  - Red Hat
  - CentOS

Lab 1.1

- Repeat the procedure “Performing a Manual Installation” to install two more servers. Details about the additional configuration on these servers follow in exercises in later chapters. For now, it is sufficient to ensure that the following conditions are met:
- Use the server names server1 and server2.
- Set the obtain an IP address automatically.
- Make sure to keep at least 1GB of disk space as unallocated disk space (which is not assigned to any partition) so that you have free space to work on the partitioning exercises in later chapters.
- Install one server using the Minimal installation pattern, and another server using the Server with GUI installation pattern.

Chapter 2:

Using Essential Tools

Chapter 2 Objectives

- The following topics are covered in this chapter:
  - Basic Shell Skills
  - Understanding the Shell Environment
  - Editing Files with vim
  - Finding Help

- The following RHCSA exam objectives are covered in this chapter:
  - Use input-output redirection
  - Create and edit text files
  - Locate, read, and use system documentation including man, info, and files in /usr/share/doc

Basic Shell Skills

- The shell is the default working environment for a Linux administrator. It is the environment where users and administrators enter commands that are executed by the operating system. Different shells for Linux are available, but bash is the common shell. So when we are talking about “the shell” in this book, we are actually talking about the *bash* shell. This chapter provides an overview of some of the items that you will encounter when working with the shell.

Executing Commands

- The purpose of the Linux shell is that it provides an environment in which commands can be executed. The shell takes care of interpreting the command that a user has entered correctly. To do this, the shell makes a difference between three kinds of commands:

- Aliases
- Internal commands
- External commands

Executing Commands

- An alias is a command that a user can define as needed. Some aliases are provided bydefault; type **alias** on the command line to get an overview.

- To define an alias, use

alias newcommand=’oldcommand’ ,

- as in the default alias

ll=’ls -l --color=auto’ .

- Aliases are executed before anything else.

Executing Commands

- An internal command is a command that is a part of the shell itself. It is available when the shell is loaded and can be executed from memory without any lookup from disk. An external command is a command that exists as an executable file on disk of the computer. Because it has to be read from disk, it is a bit slower.
- When a user executes a command, the shell first looks to determine whether it is an internal command; if it is not, it looks for an executable file with a name that matches the command on disk.
- To find out whether a command is a bash internal, or an executable file on disk, you can use the **type** command

Executing Commands

- To look up external commands, the **$PATH** variable is used. This variable defines a list of directories that is searched for a matching filename when a user enters a command. To find out which exact command the shell will be using, you can use the **which** command. For instance, type **which ls** to find out where the shell will get the **ls** command from.

Executing Commands

- You should notice that for security reasons that the current directory is not in the **$PATH** variable and that Linux does not look in the current directory to see whether a specific command is available from that directory. That is why you need to start a command that is in the current directory but nowhere in the **$PATH** by including **./** in front of it. The dot stands for the current directory, and by running it as **./** , you'll tell bash to look for the command in the current directory.
- The **$PATH** variable can be set for specific users, but in general, most users will be using the same PATH variable. The only exception to this is the user root, who needs access to specific administration commands. In Exercise 2.1 , you learn some of the basics about working with commands.

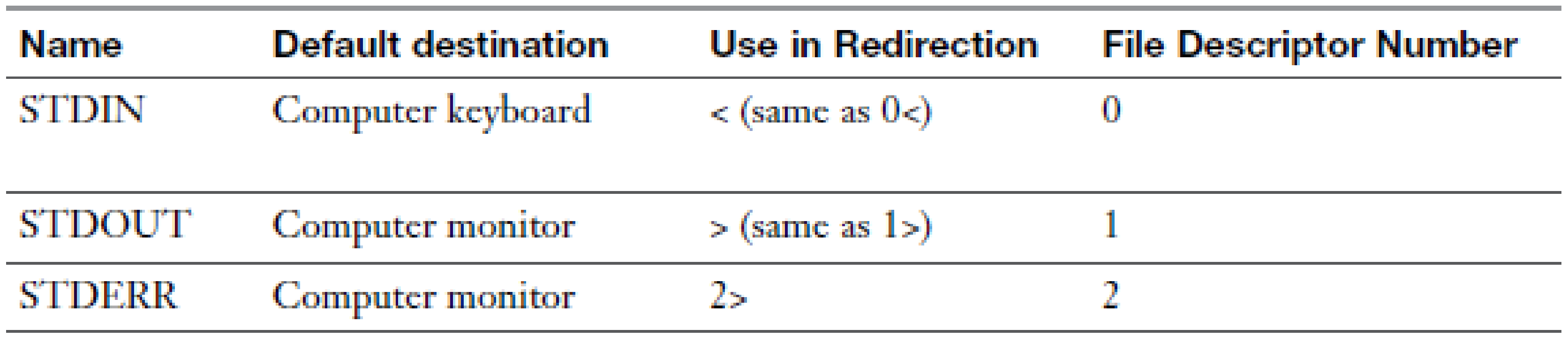
Exercise 2.1 Using Internal and External Commands from the Shell

1. Authenticate as the user user who you created in Chapter 1 , “Installing Red Hat Enterprise Linux Server,” when installing your server.
2. Type **time ls** . This executes the **ls** command where the bash internal **time** shows information about the time it took to complete this command.
3. Now type **which time** . This shows the filename /bin/time that was found in the $PATH variable.
4. Type **echo $PATH** to show the contents of the $PATH variable. You can see that /bin is included in the list, but because there also is an internal command time , the time command from the path will not be executed unless you tell the shell specifically to do so.
5. Type **/bin/time ls** to run the **/bin/time** command when executing **ls** . You’ll notice that the output differs completely.

I/O Redirection

- By default when a command is executed it shows its results on the screen of the computer you are working on. The computer monitor is the so-called standard output, which is also referred to as the **STDOUT**. The shell also has default destinations to send error messages to and to accept input. Table 2.2 gives an overview of all three.

Table 2.2 Standard Input, Output, and Error Overview



Name	Default destination	Use in Redirection	File Descriptor Number
STDIN	Computer keyboard	< (same as 0<)	0
STDOUT	Computer monitor	> (same as 1>)	1
STDERR	Computer monitor	2>	2

Run the command **find / -name “*.rpm”**
find / -name “*.rpm” 2> /dev/null
find / -name “*.rpm” > rpm-result.txt 2> /dev/null
find / -name “*.rpm” >> rpm-result.txt 2> /dev/null

I/O Redirection

- In I/O redirection, files can be used to replace the default **STDIN**, **STDOUT**, and **STDERR**. You can also redirect to *device files* . A device file on Linux is a file that is used to access specific hardware. Your hard disk for instance can be referred to as **/dev/sda**, the console of your server is known as **/dev/console** or **/dev/tty1**, and if you want to discard a commands output, you can redirect to **/dev/null**. Notice that to access most device files you need to be root.

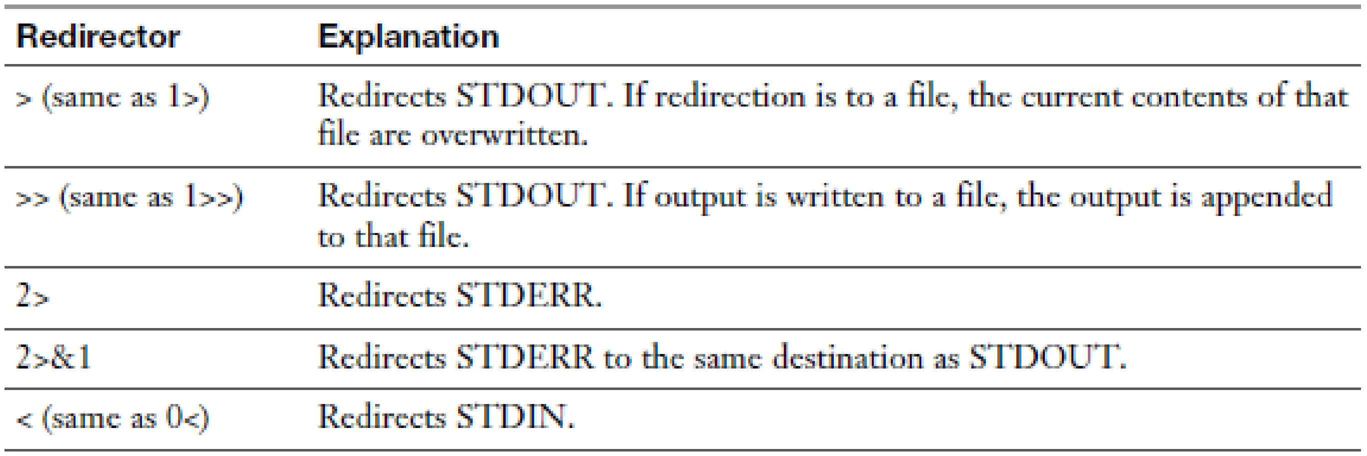


Table 2.3 Common Bash Redirectors

Redirector	Explanation
> (same as 1>)	Redirects STDOUT. If redirection is to a file, the current contents of that file are overwritten.
>> (same as 1>>)	Redirects STDOUT. If output is written to a file, the output is appended to that file.
2>	Redirects STDERR.
2>&1	Redirects STDERR to the same destination as STDOUT.
< (same as 0<)	Redirects STDIN.

Using Pipes

- Where an I/O redirector is used to use alternatives for keyboard and computer monitor, a pipe can be used to catch the output of one command and use that as input for a second command.
- If a user runs the command **ls** , for instance, the output of the command is shown onscreen. If the user uses **ls | less** , the commands **ls** and **less** are started in parallel. The *standard output* of the **ls** command is connected to the *standard input* of **less**.

Exercise 2.2 Using I/O Redirection and Pipes

1. Open a shell as user user and type **cd** without any arguments. This ensures that the home directory of this user is the current directory while working on this exercise. Type **pwd** to verify this.
2. Type **ls.** You'll see the results onscreen
3. Type **ls > /dev/null** . This redirects the STDOUT to the null device, with the result that you will not see it.
4. Type **ls ilwehgi > /dev/null** . This command shows a "no such file or directory" message onscreen. You see the message because it is not STDOUT, but an error message that is written to STDERR.
5. Type **ls ilwehgi 2> /dev/null** . Now you will not see the error message anymore.
6. Type **ls ilwehgi Documents 2> /dev/null** . This shows the name of the Documents folder in your home directory while hiding the error message.
7. Type **ls ilwehgi Documents 2> /dev/null > output** . In this command, you still write the error message to /dev/null while sending STDOUT to a file with the name output that will be created in your home directory.
8. Type **cat output** to show the contents of this file.

Exercise 2.2 Using I/O Redirection and Pipes

9. Type **echo hello > output** . This overwrites the contents of the output file.
10. Type **ls >> output** . This appends the result of the **ls** command to the output file.
11. Type **ls -R /**. This shows a long list of files and folders scrolling over your computer monitor. (You may want to type **Ctrl+C** to stop [or wait a long time]).
12. Type **ls -R | less** . This shows the same result, but in the pager **less** , where you can scroll up and down using the arrow keys on your keyboard.
13. Type **q** to close **less** . This will also end the **ls** program.
14. Type **ls > /dev/tty1** . This gives an error message because you are executing the command as an ordinary user (unless you were logged in to tty1). Only the user root has permission to write to device files directly.

History

- A convenient feature of the bash shell is the bash history. Bash is configured to keep the last 1,000 commands you have used (and if shell session is never closed, the exact number can grow even much beyond that). When a shell session is closed, the history of that session is updated to the history file. The name of this file is **.bash_history**, and it is created in the home directory of the user who started a specific shell session. Notice that the history file is closed only when the shell session is closed; until that moment, all commands in the history are kept in memory.

History

- The history feature makes it easy to repeat complex commands. There are several ways of working with history:
  - Type **history** to show a list of all commands in the bash history.
  - Use **Ctrl+R** to open the prompt from which you can do backward searches in commands that you have previously used. Just type a part of the command you are looking for, and it will be displayed automatically. Use **Ctrl+R** to search further backward based on the same search criteria.
  - Type **!number** to execute a command with a specific number from history.
  - Type **!sometext** to execute the last command that starts with sometext. *Notice that this is a potentially dangerous command because the command that was found is executed immediately!*

Exercise 2.3 Working with History

1. Make sure that you have opened a shell as user user.
2. Type **history** to get an overview of commands that you have previously used.
3. Type some commands, such as the following:

   ```
   ls
   pwd
   cat /etc/hosts
   ls -l.
   ```

   The goal is to fill the history a bit.
4. Open a second terminal on your server by right-clicking the graphical desktop and selecting the **Open in Terminal** menu option.
5. Type **history** from this second terminal window. Notice that you do not see the commands that you just typed in the other terminal. That is because the history file has not been updated yet.

Exercise 2.3 Working with History

6. From the first terminal session, type **Ctrl+R**. From the prompt that opens now, type **ls**. You'll see the last **ls** command you used. Press **Ctrl+R** again. You'll now see that you are looking backward and that the previous **ssh** command is highlighted. Press **Enter** to execute it.
7. Type **history | grep cat**. The **grep** command searches the history output for any commands that contained the text *cat*. Remember the command number of one of the **cat** commands you have previously used.
8. Type **!nn**, where *nn* is replaced by the number you remembered in Step 7. You'll see that the last **cat** command is repeated.
9. Close this terminal by typing **exit**.
10. From the remaining terminal window, type **history -c**. This wipes all history that is currently in memory. Close this terminal session as well.
11. Open a new terminal session and type **history**. It may be a bit unexpected, but you'll see a list of commands anyway. That is because **history -c** clears the in-memory history, but it does not remove the .bash_history file in your home directory.
12. As an alternative to deleting the history file, you can also use **history -w** after using **history -c**.

Bash Completion

- Another useful feature of the bash shell is automatic completion. This feature helps you in finding the command you need, and it also works on variables and filenames, and on some occasions even within command shells that are opened.
- Just type the beginning of a command and press the Tab key on your computer's keyboard. If there is only one option for completion, bash will complete the command automatically for you. If there are several options, you need to press the Tab key once more to get an overview of all the available options.

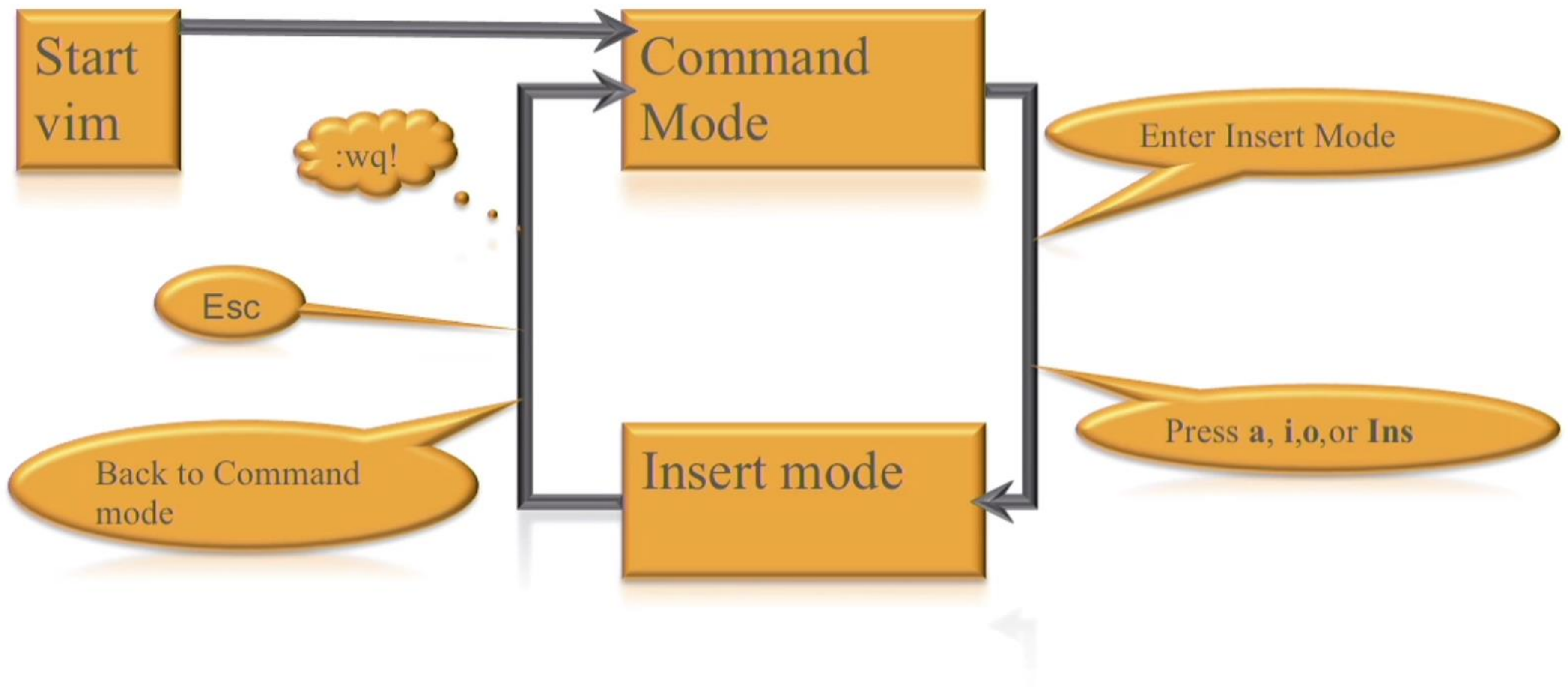
Exercise 2.4 Using Bash Completion

1. Still from a user shell, type **gd** and press **Tab**. You'll see that nothing happens.
2. Press **Tab** again. Bash now shows a short list of all commands that start with the letters *gd*.
3. To make it clear to bash what you want, type **i** (so that your prompt at this point shows a command **gdi**. Press **Tab** again. Bash now knows what you want and opens gdisk for you. Press **Enter** to close the prompt that was just opened.
4. Use **cd /etc** to go to the /etc directory.
5. Type **cat pas[Tab]**. Because there is one file only that starts with *pas*, bash knows what to do and automatically completes the filename. Press **Enter** to execute the command.

Editing Files with *vim*

- Over the years, many text editors have been created for Linux. One editor really matters, though, and that is vi. Even if some other text editors are easier to use, vi is the only text editor that is always available. That is why as a Linux administrator you need to know how to work with vi. Only one alternative is permitted, and that is vim.
- Vim is “**vi improved”** it is a complete rewrite of vi with a lot of enhancements that make working with vi easier, such as syntax highlighting for many configuration files which makes it easy to recognize typing errors that you have made. All that you learn in this section about vim works on vi as well.

Editing Files with *vim*



Editing Files with *vim*

vim command	explanation
Esc	Switches from input mode to command mode. Use this before typing any command.
i, a	Switches from command mode to input mode at (i) or after (a) the current cursor position.
o	Opens a new line below the current cursor position and goes to input mode.
:wq	Writes the current file and quits.
:q!	Quits the file without applying any changes. The ! forces the command to do its work. Only add the ! if you really know what you are doing.
:w filename	Writes the current file with a new filename.
dd	Deletes the current line.
yy	Copies the current line.
p	Pastes the current selection.
v	Enters visual mode, which allows you to select a block of text using the arrow keys. Use **d** to cut, or **y** to copy the selection.
u	Undoes the last command. Repeat as often as necessary.
Ctrl+r	Redoes the last undo.
gg	Goes to the first line in the document.
G	Goes to the last line in the document.

Editing Files with *vim*

vim command	explanation
/text	Searches for *text* from the current cursor position forward.
?text	Searches for *text* from the current cursor position backward.
^	Goes to the first position in the current line.
$	Goes to the last position in the current line.
!ls	Adds the output of **ls** (or any other command) in the current file.
:%s/old/new/g	Replaces all occurrences of *old* with *new*.

Exercise 2.5 Vim Practice

1. Type **vim ~/testfile**. This starts vim and opens a file with the name testfile in ~, which represents your current home directory.
2. Press **i** to enter input mode and type the following text

   ```
   cow
   sheep
   ox
   chicken
   snake
   fish
   oxygen
   ```

3. Press **Esc** to get back to command mode and type **:w** to write the file using the same filename.
4. Type **:3** to go to line number 3.
5. Type **dd** to delete this line.
6. Type **dd** again to delete another line.
7. Type **u** to undo the last deletion.
8. Type **o** to open a new line.
9. Enter some more text at the current cursor position:

   ```
   tree
   farm
   ```

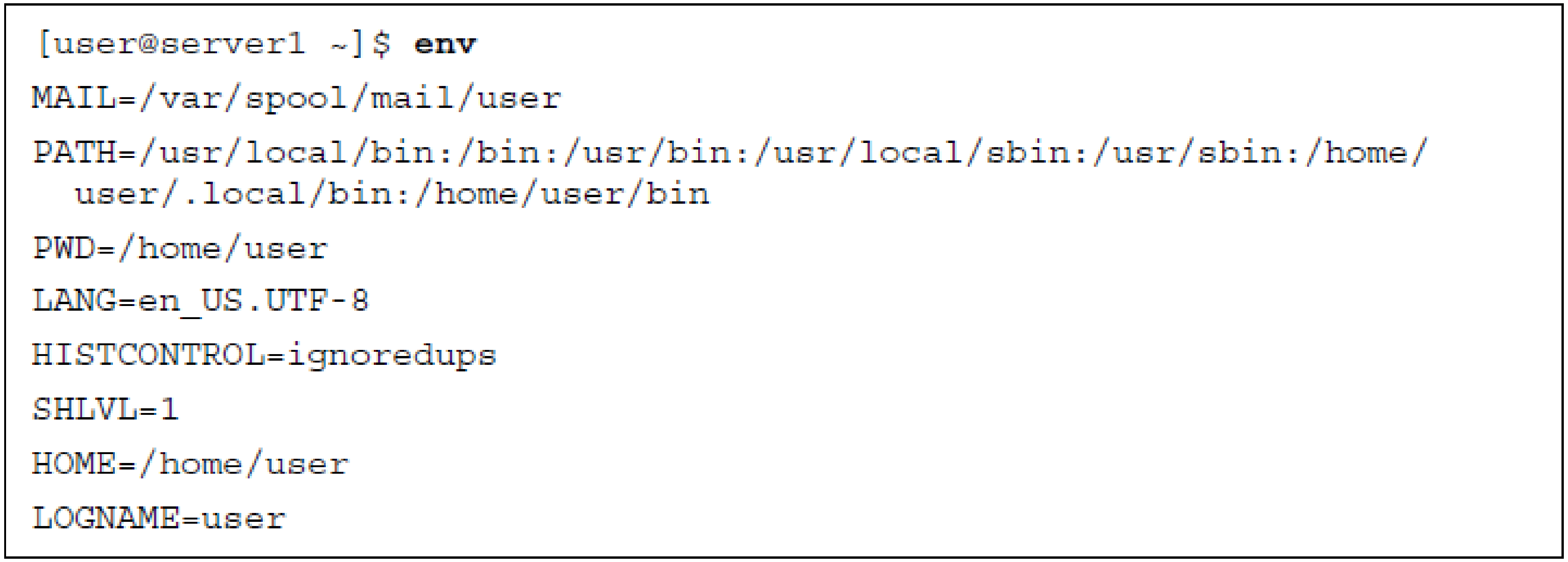
Exercise 2.5 Vim Practice

10. Press **Esc** to get back into command mode.
11. Type **:%s/ox/OX/g.**
12. Type **:wq** to write the file and quit. If for some reason that does not work, use **:wq!**.

Understanding the Shell Environment

- When you are working from a shell, an environment is created to ensure that all that is happening is happening the right way. This environment consists of variables that define the user environment, such as the **$PATH** variable discussed earlier. In this section, you get a brief overview of the shell environment and how it is created.

Understanding Variables


Listing 2.1 Displaying the Current Environment

```
[user@server1 ~]$ env
MAIL=/var/spool/mail/user
PATH=/usr/local/bin:/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/
  user/.local/bin:/home/user/bin
PWD=/home/user
LANG=en_US.UTF-8
HISTCONTROL=ignoredups
SHLVL=1
HOME=/home/user
LOGNAME=user
```

To read the value of a variable, a user can use the **echo** command, followed by the name of the variable, as in **echo $PATH** , which reads the current value of the PATH variable and prints that on the STDOUT.

Environment Configuration Files

- When a user logs in, an environment is created for that user automatically. This happens based on four different files where some script code can be specified and where variables can be defined for use by one specific user:

  - **/etc/profile:** This is the generic file that is processed by all users upon login.
  - **/etc/bashrc:** This file is processed when subshells are started.
  - **~/.bash_profile:** In this file, user-specific login shell variables can be defined.
  - **~/.bashrc:** In this user-specific file, subshell variables can be defined.

Using */etc/motd* and */etc/issue*

- Bash offers an option to include messages in the **/etc/motd** and the **/etc/issue** files.
- Messages in **/etc/motd** display after a user has successfully logged in to a shell. (Notice that users in a graphical environment do not see its contents after a graphical login.)
- Using **/etc/motd** can be a convenient way for system administrators to inform users.
- Another way to send information to users is by using **/etc/issue**. The contents of this file display before the user logs in. This provides an excellent means of specifying specific login instructions to users who are not logged in yet.

Exercise 2.6 Managing the Shell Environment

1. Open a shell in which you are user.
2. Type **echo $LANG** to show the contents of the variable that sets your system keyboard and language settings.
3. Type **ls --help**. You'll see that help about the **ls** command is displayed in the current language settings of your computer.
4. Type **LANG=fr_FR.UTF-8**. This temporarily sets the language variable to French.
5. Type **ls --help** again. You'll see that now the ls help text is displayed in French.
6. Type **exit** to close your terminal window. Because you have not changed the contents of any of the previously mentioned files, while opening a new shell the original value of the LANG variable will be used.

Exercise 2.6 Managing the Shell Environment

7. Open a shell as user again.
8. Verify the current value of the LANG variable by typing echo **$LANG**.
9. Use **vim .bashrc** to open the .bashrc configuration file.
10. In this file, add the line **COLOR=red** to set a variable with the name COLOR and assign it the value red.
11. Close the user shell and open a new user shell.
12. Verify that the variable COLOR has been set, by using **echo $COLOR**. Because the .bashrc file is included in the login procedure, the variable is set after logging in.

Finding Help

- On an average Linux system, hundreds of commands are available (way too many to ever be able to remember all of them, which is why using the help resources on your computer is so very important). This section provides a brief overview about using man, and (more important) what you can do with man if you do not know the exact command you are looking for.

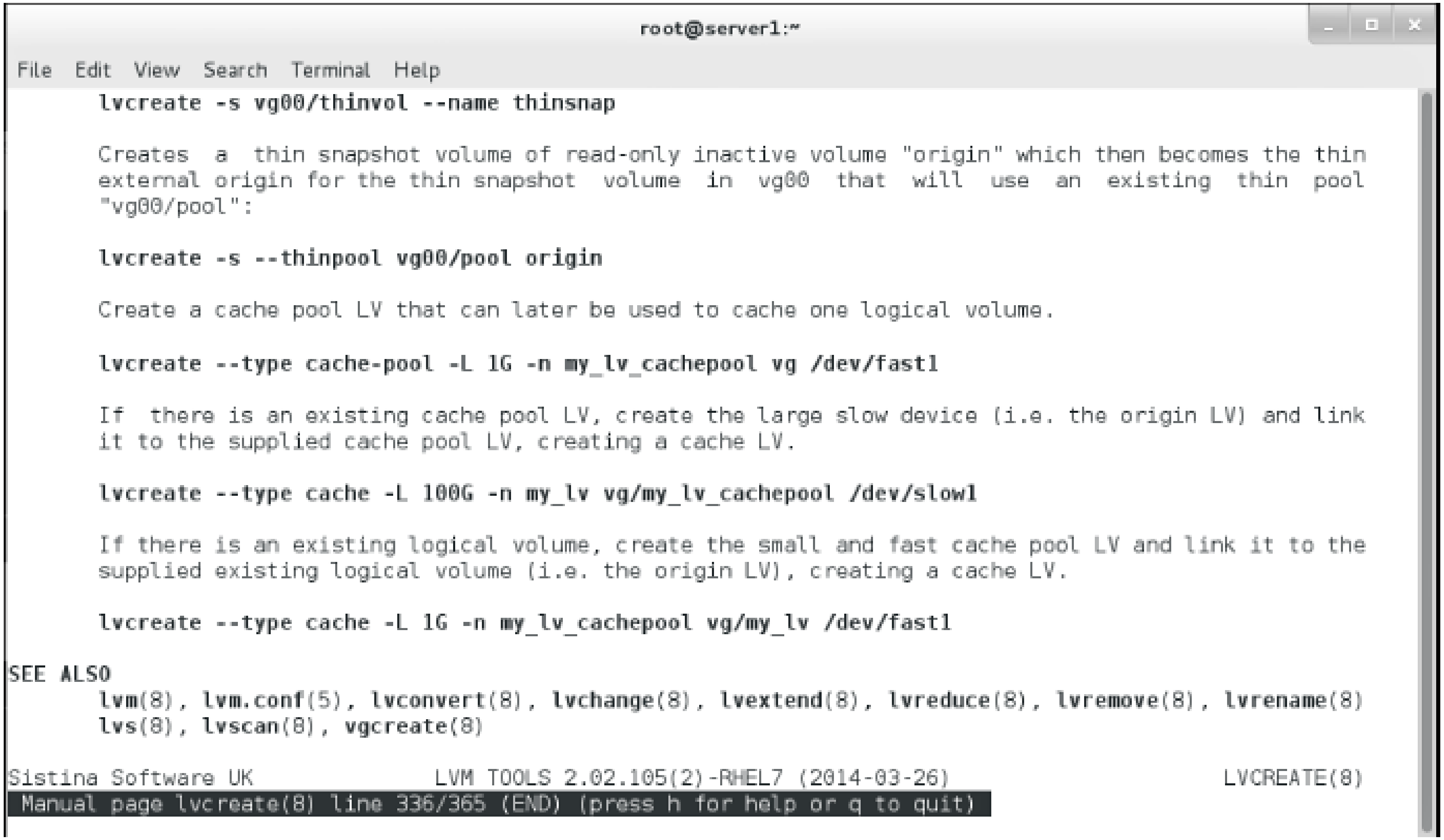
Using *--help*

- The quickest way to get an overview of how to use a command is by running the command with the **--help** option. Nearly all commands will display a usage summary when using this option. The list of options that is shown in this way is of use mainly when you already have a generic understanding of how to use the command and need a quick overview of options available with the command.

Using *man*

- When using the Linux command line, you will at some point consult man pages. Man is what makes working from the command line doable. If you do not know how a command is used, the man page of that command will provide valuable insight. This section covers a few man essentials.
- To start with, the most important parts of the man page in general are at the bottom of the man page. Here you'll find two important sections: In many cases there are examples; if there are no examples, there is always a "see also" section. This brings you to related man pages, which is useful if you have just not hit the right man page. To get to the bottom of the man page as fast as possible, use the **G** command. You can also type **/example** to search the man page for any examples. Figure 2.1 shows what the end of a man page may look like.

Using *man*



```
       lvcreate -s vg00/thinvol --name thinsnap

       Creates  a  thin snapshot volume of read-only inactive volume "origin" which then becomes the thin
       external origin for the thin snapshot  volume  in  vg00  that  will  use  an  existing  thin  pool
       "vg00/pool":

       lvcreate -s --thinpool vg00/pool origin

       Create a cache pool LV that can later be used to cache one logical volume.

       lvcreate --type cache-pool -L 1G -n my_lv_cachepool vg /dev/fast1

       If  there is an existing cache pool LV, create the large slow device (i.e. the origin LV) and link
       it to the supplied cache pool LV, creating a cache LV.

       lvcreate --type cache -L 100G -n my_lv vg/my_lv_cachepool /dev/slow1

       If there is an existing logical volume, create the small and fast cache pool LV and link it to the
       supplied existing logical volume (i.e. the origin LV), creating a cache LV.

       lvcreate --type cache -L 1G -n my_lv_cachepool vg/my_lv /dev/fast1

SEE ALSO
       lvm(8), lvm.conf(5), lvconvert(8), lvchange(8), lvextend(8), lvreduce(8), lvremove(8), lvrename(8)
       lvs(8), lvscan(8), vgcreate(8)

Sistina Software UK            LVM TOOLS 2.02.105(2)-RHEL7 (2014-03-26)                    LVCREATE(8)
 Manual page lvcreate(8) line 336/365 (END) (press h for help or q to quit)
```

Figure 2.1 Sample man page contents.

Finding the Right *man* Page

- To find information in man pages, you can search the mandb database by using **apropos** or **man -k** . If the database is current, getting access to the information you need is easy. Just type **man -k** , followed by the keyword you want to search for. This command looks in the summary of all man pages that are stored in the mandb database. Listing 2.2 shows a partial result of this command.
- Instead of using **man -k** , you can use the **apropos** command. This command is equivalent to **man -k** .

Finding the Right *man* Page

Listing 2.2 Searching man Pages with **man -k**

```
[root@server1 ~]# man -k partition
addpart (8)          - simple wrapper around the "add partition" ioctl
cfdisk (8)           - display or manipulate disk partition table
cgdisk (8)           - Curses-based GUID partition table (GPT) manipulator
delpart (8)          - simple wrapper around the "del partition" ioctl
fdisk (8)            - manipulate disk partition table
fixparts (8)         - MBR partition table repair utility
gdisk (8)            - Interactive GUID partition table (GPT) manipulator
iostat (1)           - Report Central Processing Unit (CPU) statistics and in...
kpartx (8)           - Create device maps from partition tables
mpartition (1)       - partition an MSDOS hard disk
os-prober (1)        - Discover bootable partitions on the local system.
partprobe (8)        - inform the OS of partition table changes
partx (8)            - tell the Linux kernel about the presence and numbering...
pvcreate (8)         - initialize a disk or partition for use by LVM
pvresize (8)         - resize a disk or partition in use by LVM2
resizepart (8)       - simple wrapper around the "resize partition" ioctl
sfdisk (8)           - partition table manipulator for Linux
sgdisk              (- Command-line GUID partition table (GPT) manipulator fo...
systemd-efi-boot-generator (8) - Generator for automatically mounting the EFI...
systemd-gpt-auto-generator (8) - Generator for automatically discovering and ..
```

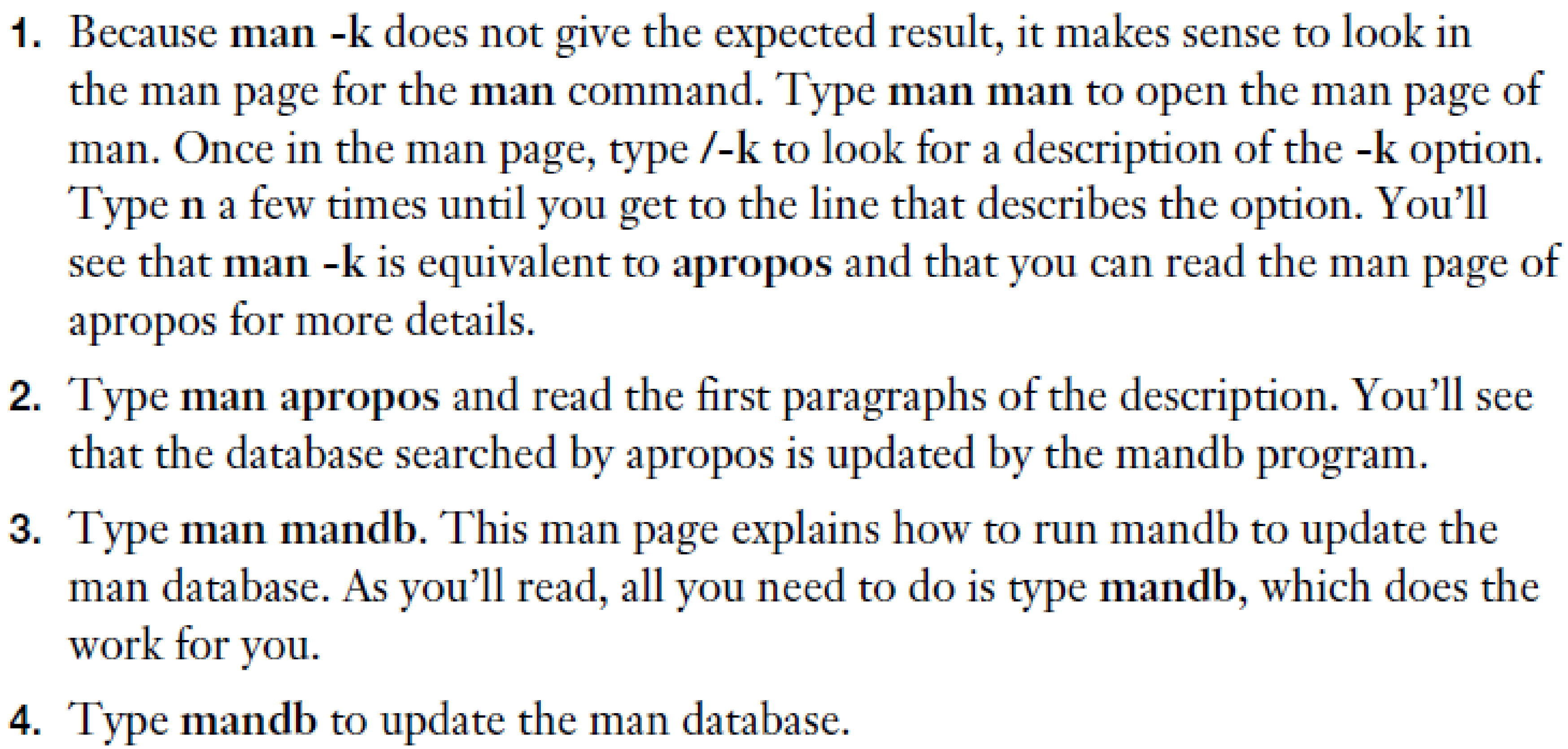
Finding the Right *man* Page

- Man pages are categorized in different sections. The most relevant sections for system administrators are as follows
  - 1: Executable programs or shell commands
  - 5: File formats and conventions
  - 8: System administration commands

- So, if you are looking for the configuration file that has something to do with passwords,
  use **man -k password | grep 5**

- Or if you are looking for the command that an administrator would use to create partitions,
  use **man -k partition | grep 8** .

Exercise 2.7 Using *man -k*



1. Because **man -k** does not give the expected result, it makes sense to look in the man page for the **man** command. Type **man man** to open the man page of man. Once in the man page, type **/-k** to look for a description of the **-k** option. Type **n** a few times until you get to the line that describes the option. You'll see that **man -k** is equivalent to **apropos** and that you can read the man page of apropos for more details.
2. Type **man apropos** and read the first paragraphs of the description. You'll see that the database searched by apropos is updated by the mandb program.
3. Type **man mandb**. This man page explains how to run mandb to update the man database. As you'll read, all you need to do is type **mandb**, which does the work for you.
4. Type **mandb** to update the man database.

Using *info*

- Apart from the information that you'll find in man, another system provides help about command usage. This is the info system.
- Most commands are documented in man, but some commands are documented only in the info system. If that is the case, the "see also" section of the man page of that command will tell you that "The full documentation for ... is maintained as a Texinfo manual." If that happens, you can read the Info page using the command **pinfo** or **info** . Both commands work, but in **pinfo** , special items such as menu items are clearly indicated, which is why using **pinfo** is recommended.

Exercise 2.8 Using *info*

1. Type **man ls**. Type **G** to go to the end of the man page and look at the "see also" section. It tells you that the full documentation for ls is maintained as a Texinfo manual. Quit the man page by pressing **Q**.
2. Type **pinfo coreutils 'ls invocation'**. This shows the information about ls usage in the pinfo pager. Read through it and press **Q** when done.

Using */usr/share/doc* Documentation Files

- A third source of information consists of files that are sometimes copied to the /usr/share/doc directory. This happens in particular for services and larger systems that are a bit more complicated. You will not typically find much information about a command like **ls** , but some services do provide useful information in /usr/share/doc.

- Some services store very useful information in this directory, like rsyslog, bind, Kerberos, and OpenSSL. For setting up advanced services, you need access to this information. For setting up services on the RHCSA and RHCE exams, you can probably do without.

Summary

- In this chapter, you read about essential Linux administration tasks. You learned about some of the important shell basics, such as I/O redirection, working with history, and management of the environment. You also learned how to edit text files with the vim editor. In the last part of this chapter, you learned how to find information using **man** and related commands.

Define Key Terms

- Define the following key terms:
  - Shell
  - Bash
  - internal command
  - external command
  - $PATH
  - Variable
  - STDIN
  - STDOUT
  - STDERR
  - Pipe
  - Environment
  - login shell
  - subshell

Lab 2.1

1. Modify your shell environment so that on every subshell that is started a variable is set. The name of the variable should be COLOR, and the value should be set to red. Verify that it is working.

2. Use the appropriate tools to find the command that you can use to set the system time 1 minute ahead.

Chapter 3:

Essential File Management Tools

Chapter 3 Objectives

- The following topics are covered in this chapter:
  - Working with the File System Hierarchy
  - Managing Files
  - Using Links
  - Working with Archives and Compressed Files

- The following RHCSA exam objectives are covered in this chapter:
  - Archive, compress, unpack, and uncompress files
  - Create, delete, copy, and move files directories
  - Create hard and soft links

Working with the File System Hierarchy

- To manage a Linux system, you should be familiar with the default directories that exist on almost all Linux systems. This section describes these directories and explains how mounts are used to compose the file system hierarchy.

Defining the File System Hierarchy

- The file system on most Linux systems is organized in a similar way. The layout of the Linux file system is defined in the Filesystem Hierarchy Standard (FHS), and this file system hierarchy is described in **man 7 hier** .
- Table 3.2 shows an overview of the most significant directories that you'll encounter on a Red Hat Enterprise Linux (RHEL) system, as specified by the FSH.

Defining the File System Hierarchy

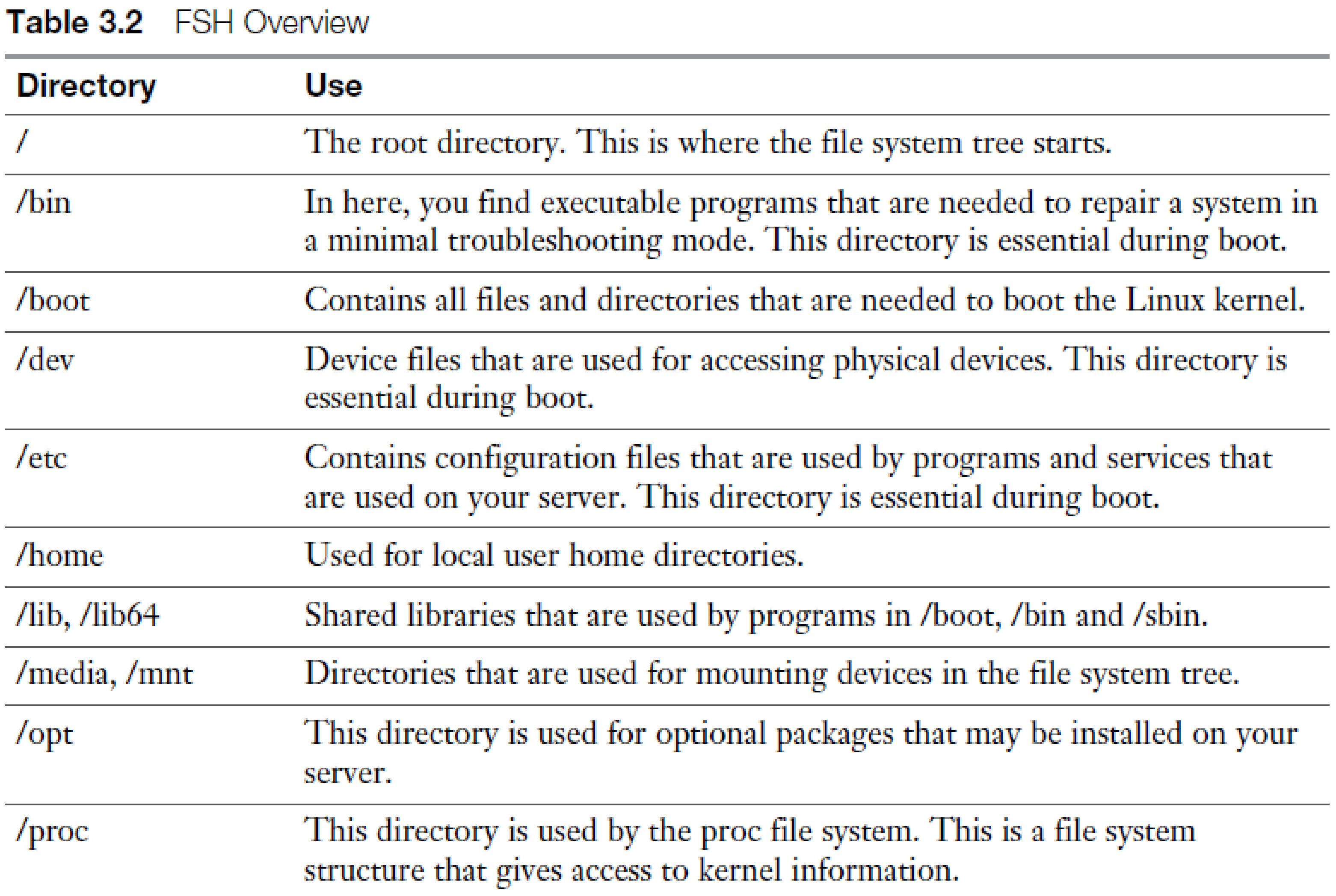
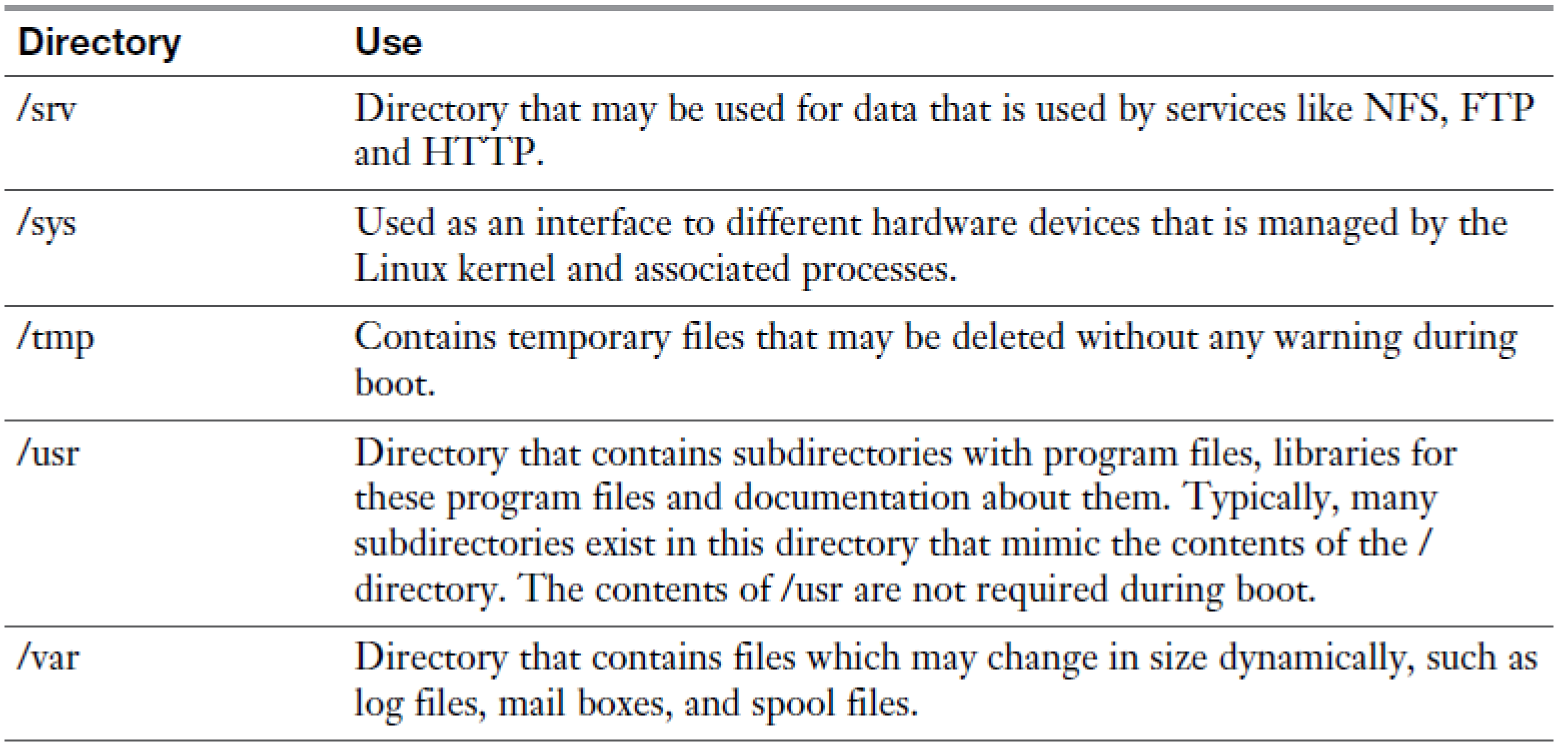


Table 3.2 FSH Overview

Directory	Use
/	The root directory. This is where the file system tree starts.
/bin	In here, you find executable programs that are needed to repair a system in a minimal troubleshooting mode. This directory is essential during boot.
/boot	Contains all files and directories that are needed to boot the Linux kernel.
/dev	Device files that are used for accessing physical devices. This directory is essential during boot.
/etc	Contains configuration files that are used by programs and services that are used on your server. This directory is essential during boot.
/home	Used for local user home directories.
/lib, /lib64	Shared libraries that are used by programs in /boot, /bin and /sbin.
/media, /mnt	Directories that are used for mounting devices in the file system tree.
/opt	This directory is used for optional packages that may be installed on your server.
/proc	This directory is used by the proc file system. This is a file system structure that gives access to kernel information.

Defining the File System Hierarchy



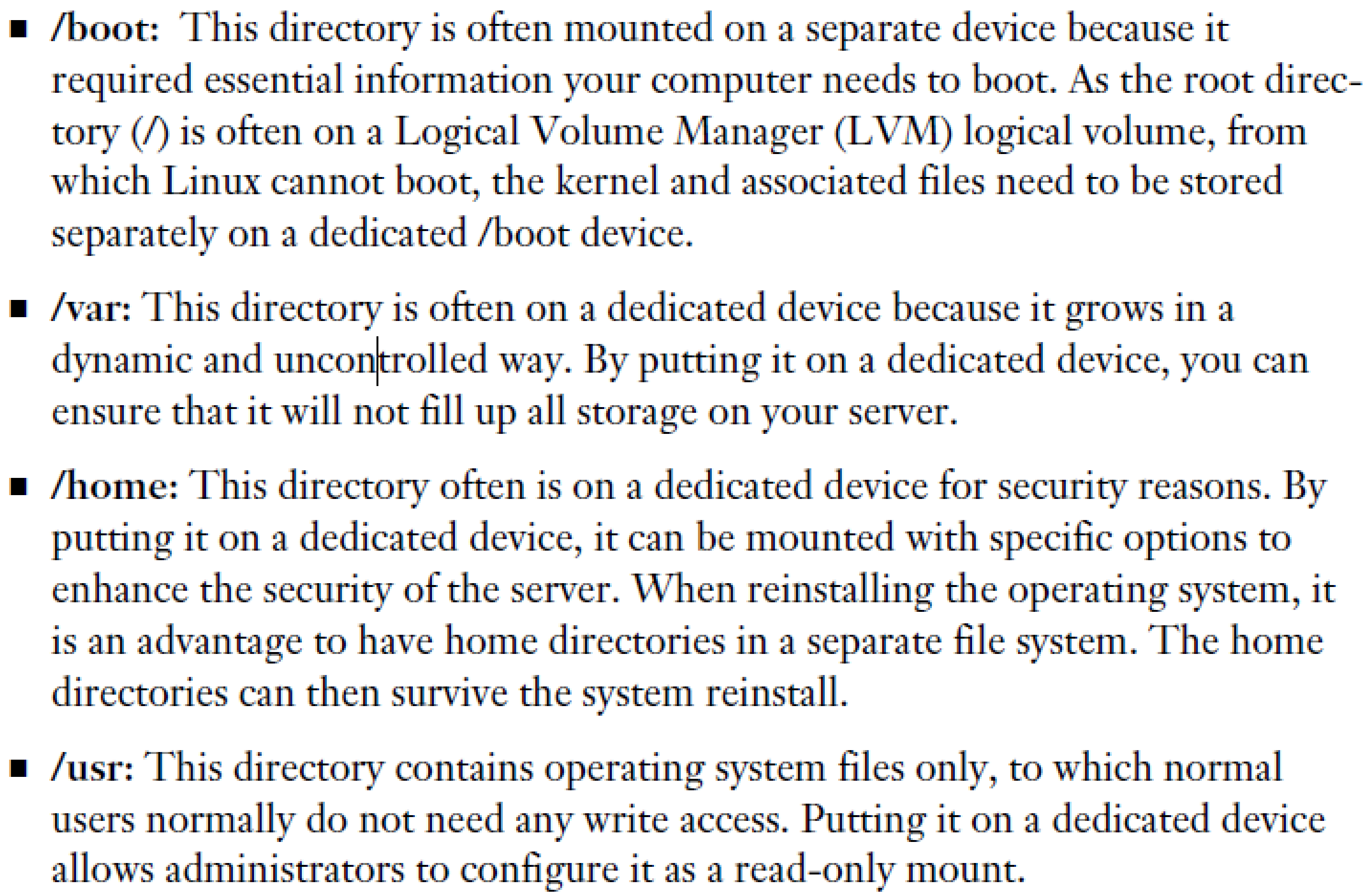
Directory	Use
/srv	Directory that may be used for data that is used by services like NFS, FTP and HTTP.
/sys	Used as an interface to different hardware devices that is managed by the Linux kernel and associated processes.
/tmp	Contains temporary files that may be deleted without any warning during boot.
/usr	Directory that contains subdirectories with program files, libraries for these program files and documentation about them. Typically, many subdirectories exist in this directory that mimic the contents of the / directory. The contents of /usr are not required during boot.
/var	Directory that contains files which may change in size dynamically, such as log files, mail boxes, and spool files.

Understanding Mounts

- To understand the organization of the Linux file system, the concept of mounting is important.
- A Linux file system is presented as one hierarchy, with the root directory (/) as its starting point. This hierarchy may be distributed over different devices and even computer systems that are mounted into the root directory.

Understanding Mounts

- Some directories are commonly mounted on dedicated devices:



- **/boot:** This directory is often mounted on a separate device because it required essential information your computer needs to boot. As the root directory (/) is often on a Logical Volume Manager (LVM) logical volume, from which Linux cannot boot, the kernel and associated files need to be stored separately on a dedicated /boot device.
- **/var:** This directory is often on a dedicated device because it grows in a dynamic and uncontrolled way. By putting it on a dedicated device, you can ensure that it will not fill up all storage on your server.
- **/home:** This directory often is on a dedicated device for security reasons. By putting it on a dedicated device, it can be mounted with specific options to enhance the security of the server. When reinstalling the operating system, it is an advantage to have home directories in a separate file system. The home directories can then survive the system reinstall.
- **/usr:** This directory contains operating system files only, to which normal users normally do not need any write access. Putting it on a dedicated device allows administrators to configure it as a read-only mount.

Understanding Mounts

- The **mount** command gives an overview of all mounted devices. To get this information, the /proc/mounts file is read, where the kernel keeps information about all current mounts. It shows kernel interfaces also, which may lead to a long list of mounted devices being displayed. Listing 3.1 shows sample output of this command.

Understanding Mounts

Listing 3.1 Sample **mount** Command Output

```
[root@server1 ~]# mount
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec,relatime,seclabel)
devtmpfs on /dev type devtmpfs (rw,nosuid,seclabel,size=929784k,nr_
  inodes=232446,mode=755)
securityfs on /sys/kernel/security type securityfs (rw,nosuid,nodev,
  noexec,relatime)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev,seclabel)
```

Understanding Mounts

- The **df -Th** command was designed to show available disk space on mounted devices; it includes most of the system mounts. Because it will look on all mounted file systems, it is a convenient command to get an overview of current system mounts. The **-h** option summarizes the output of the command in a human-readable way, and the **-T** option shows which file system type is used on the different mounts.
- The **findmnt** command shows mounts and the relation that exists between the different mounts. Because the output of the **mount** command is a bit overwhelming, you may like the output of **findmnt** . Listing 3.2 shows sample output of this command.

Understanding Mounts

Listing 3.2 Sample **findmnt** Command Output

```
[root@server1 ~]# findmnt
TARGET                                   SOURCE      FSTYPE     OPTIONS
/                                        /dev/mapper/centos-root
                            xfs           rw,relatime,seclabel,attr2,
          inode64,noquota
├─/proc                         proc       rw,nosuid,nodev,noexec,relatime
│ ├─/proc/sys/fs/binfmt_misc             systemd-1  autofs     rw,relatime,
          fd=32,pgrp=1,timeout=300,minpr
│ └─/proc/fs/nfsd                       sunrpc     nfsd       rw,relatime
├─/sys                                 sysfs      sysfs      rw,nosuid,
          nodev,noexec,relatime,seclabel
│ ├─/sys/kernel/security                securityfs
          securityf rw,nosuid,nodev,noexec,relatime
│ ├─/sys/fs/cgroup                     tmpfs      tmpfs      rw,nosuid,
          nodev,noexec,seclabel,mode=755
│ │ ├─/sys/fs/cgroup/systemd           cgroup     cgroup     rw,nosuid,
          nodev,noexec,relatime,xattr,rele
│ │ ├─/sys/fs/cgroup/cpuset            cgroup     cgroup     rw,nosuid,
          nodev,noexec,relatime,cpuset
│ │ ├─/sys/fs/cgroup/cpu,cpuacct       cgroup     cgroup     rw,nosuid,
          nodev,noexec,relatime,cpuacct,cp
│ │ ├─/sys/fs/cgroup/memory            cgroup     cgroup     rw,nosuid,
          nodev,noexec,relatime,memory
│ │ ├─/sys/fs/cgroup/devices           cgroup     cgroup     rw,nosuid,
          nodev,noexec,relatime,devices
│ │ ├─/sys/fs/cgroup/freezer           cgroup     cgroup     rw,nosuid,
          nodev,noexec,relatime,freezer
│ │ ├─/sys/fs/cgroup/net_cls           cgroup     cgroup     rw,nosuid,
          nodev,noexec,relatime,net_cls
```

Exercise 3.1 Getting an Overview of Current Mounts

- In this exercise, you use different commands to get an overview of currently mounted devices.

1. Log in as an ordinary user and type **mount** . Notice that the output of the command is quite overwhelming. If you read carefully, though, you'll see a few directories from the Linux directory structure and their corresponding mounts.

2. Now type **df -hT** . Notice that a lot fewer devices are shown. An example of the output of this command is in Listing 3.3.

Understanding Mounts

Listing 3.3 df -hT Sample Output

```
[root@server1 ~]# df -hT
    Filesystem              Type      Size  Used Avail Use% Mounted on
    /dev/mapper/centos-root xfs       5.9G  3.9G  2.1G  66% /
    devtmpfs                devtmpfs  908M     0  908M   0% /dev
    tmpfs                   tmpfs     918M  144K  917M   1% /dev/shm
    tmpfs                   tmpfs     918M   21M  897M   3% /run
    tmpfs                   tmpfs     918M     0  918M   0% /sys/fs/cgroup
    /dev/sda1               xfs       197M  131M   67M  67% /boot
```

Now that you have entered the **mount** and **df** commands, let’s have a closer look at the output of the **df -h** command.

*Note that when using the **df** command, the sizes are reported in kibibytes. The option **-m** will display these in mebibytes, and using **-h** will use a human-readable format in KiB, MiB, GiB, TiB, or PiB.*

Understanding Mounts

The output of **df** is shown in seven columns:

- **Filesystem:** The name of the device file that interacts with the disk device that is used. The real devices in the output start with /dev (which refers to the directory that is used to store device files). You can also see a couple of tmpfs devices. These are kernel devices that are used to create a temporary file system in RAM.
- **Type:** The type of file system that was used.
- **Size:** The size of the mounted device.
- **Used:** The amount of disk space the device has in use.
- **Avail:** The amount of unused disk space.
- **Use%:** The percentage of the device that currently is in use.
- **Mounted on:** The directory the device currently is mounted on.

Managing Files

- As an administrator, you need to be able to perform common file management tasks. These tasks include the following:
  - Working with wildcards
  - Managing and working with directories
  - Working with absolute and relative pathnames
  - Listing files and directories
  - Copying files and directories
  - Moving files and directories
  - Deleting files and directories

Working with Wildcards

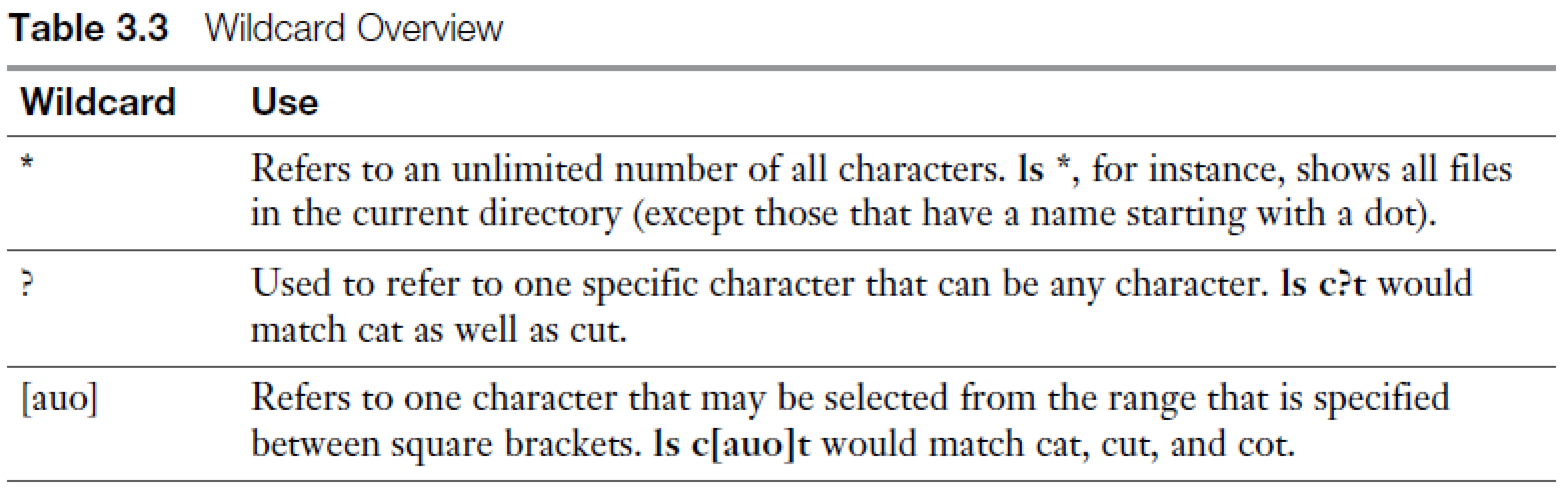


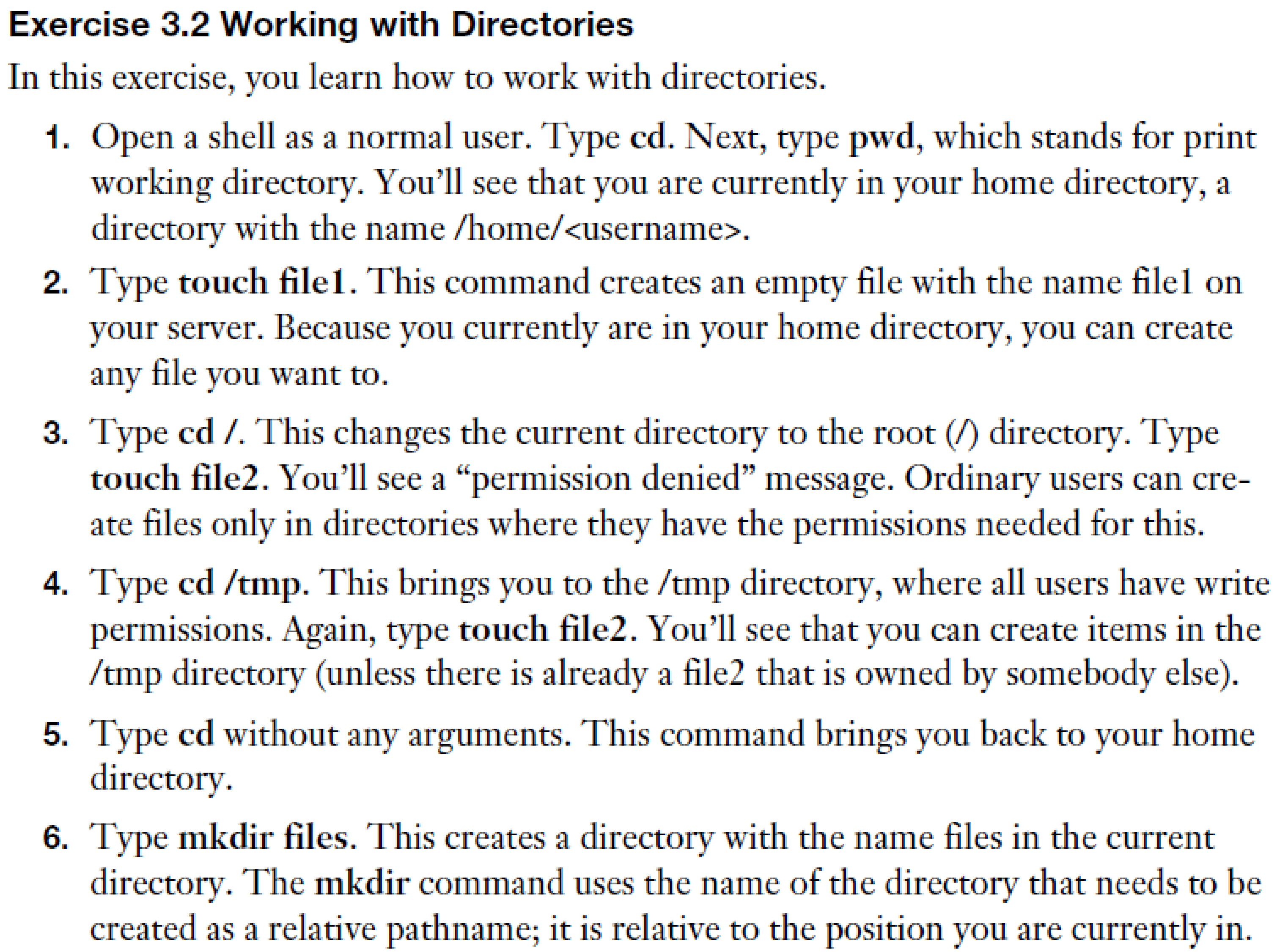
Table 3.3 Wildcard Overview

Wildcard	**Use**
*	Refers to an unlimited number of all characters. **ls ***, for instance, shows all files in the current directory (except those that have a name starting with a dot).
?	Used to refer to one specific character that can be any character. **ls c?t** would match cat as well as cut.
[auo]	Refers to one character that may be selected from the range that is specified between square brackets. **ls c[auo]t** would match cat, cut, and cot.

Managing and Working with Directories

- To organize files, Linux works with directories (also referred to as folders). You have already read about some default directories as defined by the FHS.
- When users start creating files and storing them on a server, it makes sense to provide a directory structure as well. As an administrator, you have to be able to walk through the directory structure.
- Let’s do Exercise 3.2 , in which you can discover how working with directories works.

Managing and Working with Directories



Exercise 3.2 Working with Directories

In this exercise, you learn how to work with directories.

1. Open a shell as a normal user. Type **cd**. Next, type **pwd**, which stands for print working directory. You'll see that you are currently in your home directory, a directory with the name /home/<username>.
2. Type **touch file1**. This command creates an empty file with the name file1 on your server. Because you currently are in your home directory, you can create any file you want to.
3. Type **cd /**. This changes the current directory to the root (/) directory. Type **touch file2**. You'll see a "permission denied" message. Ordinary users can create files only in directories where they have the permissions needed for this.
4. Type **cd /tmp**. This brings you to the /tmp directory, where all users have write permissions. Again, type **touch file2**. You'll see that you can create items in the /tmp directory (unless there is already a file2 that is owned by somebody else).
5. Type **cd** without any arguments. This command brings you back to your home directory.
6. Type **mkdir files**. This creates a directory with the name files in the current directory. The **mkdir** command uses the name of the directory that needs to be created as a relative pathname; it is relative to the position you are currently in.

Managing and Working with Directories

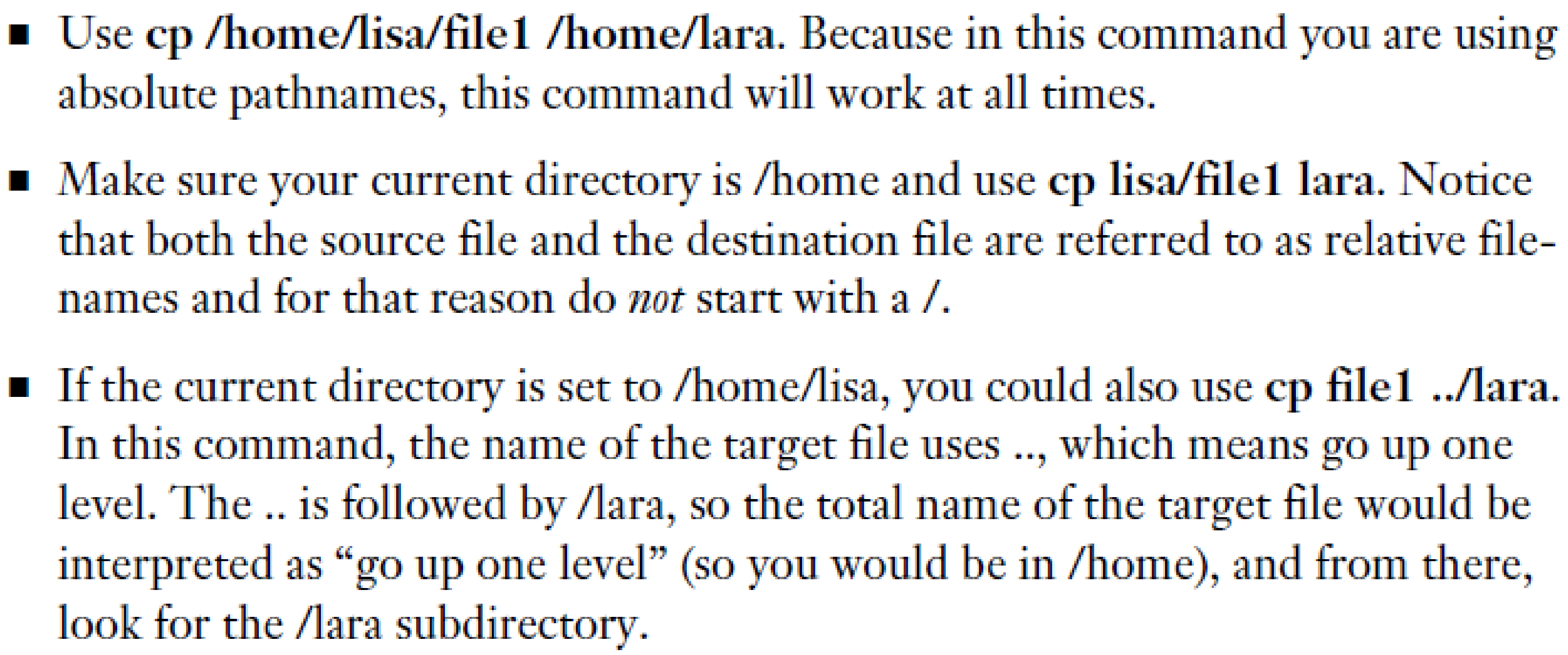
7. Type **mkdir /home/$USER/files**. In this command, you are using the variable **$USER**, which is substituted with your current username. The complete argument of **mkdir** is an absolute filename to the directory files you are trying to create. Because this directory already exist, you'll get a "file exists" error message.
8. Type **rmdir files** to remove the directory files you have just created. The **rmdir** command enables you to remove directories, but it works only if the directory is empty and does not contain any files.

Working with Absolute and Relative Pathnames

- An absolute filename, or absolute pathname, is a complete path reference to the file or directory you want to work with. This pathname starts with the root directory, followed by all subdirectories up to the actual filename. No matter what your current directory is, absolute filenames will always work. An example of an absolute filename is **/home/lisa/file1**.
- A relative filename is relative to the current directory as shown with the pwd command. It contains only the elements that are required to get from the current directory up to the item you need. Suppose that your current directory is /home (as shown by the pwd command). When you refer to the relative filename **lisa/file1**, you are referring to the absolute filename **/home/lisa/file1**.

Working with Absolute and Relative Pathnames

- When working with relative filenames, it is sometimes useful to move up one level in the hierarchy. Imagine you are logged in as root and you want to copy the file /home/lisa/file1 to the directory /home/lara. A few solutions would work:



- Use **cp /home/lisa/file1 /home/lara**. Because in this command you are using absolute pathnames, this command will work at all times.
- Make sure your current directory is /home and use **cp lisa/file1 lara**. Notice that both the source file and the destination file are referred to as relative filenames and for that reason do *not* start with a /.
- If the current directory is set to /home/lisa, you could also use **cp file1 ../lara**. In this command, the name of the target file uses .., which means go up one level. The .. is followed by /lara, so the total name of the target file would be interpreted as "go up one level" (so you would be in /home), and from there, look for the /lara subdirectory.

Listing Files and Directories

- While working with files and directories, it is useful if you can show the contents of the current directory. For this purpose, you can use the **ls** command.
- If used without arguments, **ls** shows the contents of the current directory. Some common arguments make working with **ls** easier. Table 3.4 gives an overview.

Listing Files and Directories

Table 3.4 ls Common Command-Line Options

Command	Use
ls -l	Shows a long listing, which includes information about file properties, such as creation date and permissions.
ls -a	Shows all files, including hidden files.
ls -lrt	This is a very useful command. It shows commands sorted on modification date. You'll see the most recently modified files last in the list.
ls -d	Shows the names of directories, not the contents of all directories that match the wildcards that have been used with the **ls** command.
ls -R	Shows the contents of the current directory, in addition to all of its subdirectories; that is, it **R**ecursively descends all subdirectories.

TIP A hidden file on Linux is a file that has a name that starts with a dot. Try the following: **touch .hidden**. Next, type **ls**. You will not see it. Then type **ls -a**. You'll see it.

Copying Files

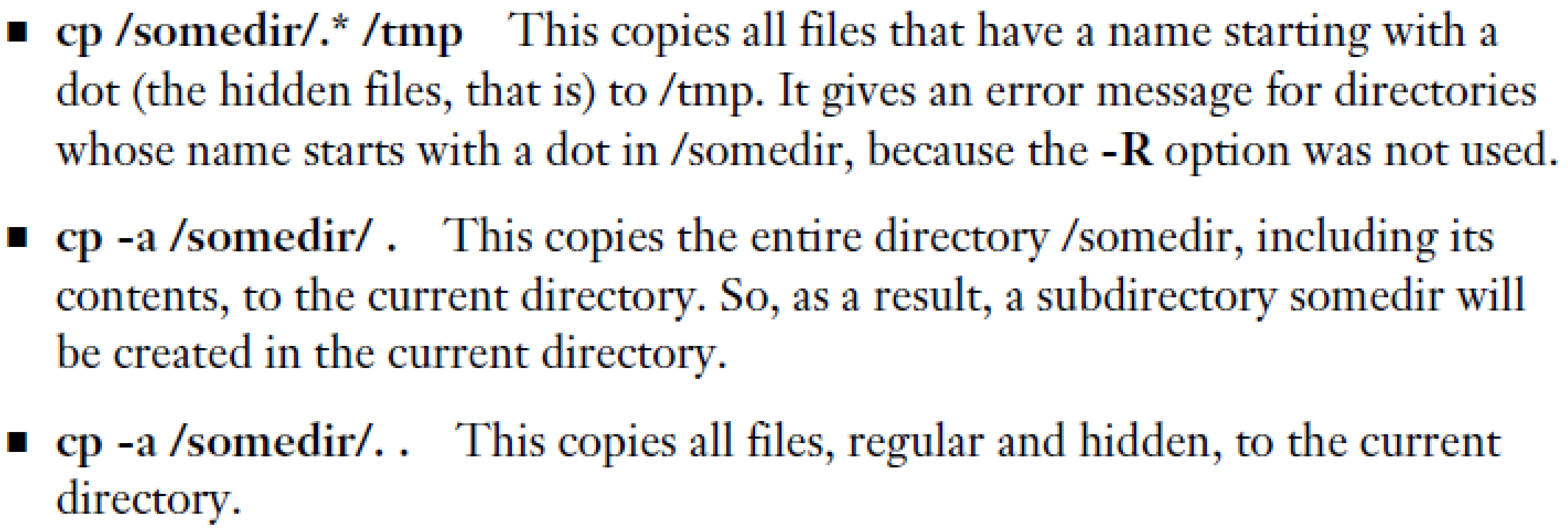
- To organize files on your server, you’ll often be copying files. The **cp** command helps you do so. Copying a single file is not difficult:
- Just use **cp /path/to/file /path/to/destination**.
  To copy the file /etc/hosts to the directory /tmp for instance, use **cp /etc/hosts /tmp**.
  This results in the file hosts being written to /tmp.
- With the **cp** command, you can also copy an entire subdirectory, with its contents and everything beneath it. To do so, use the option **-R** , which stands for recursive. (You’ll see the option **-R** with many other Linux commands also.) To copy the directory /etc and everything in it to the directory /tmp, you would, for instance, use the command **cp -R /etc /tmp** .

Copying Files

- While using the **cp** command, permissions and other properties of the files are to be considered. Without extra options, you risk permissions not being copied. If you want to make sure that you keep the current permissions, use the **-a** option, which has **cp** work in archive mode. This option ensures that permissions and all other file properties will be kept while copying.
- So, to copy an exact state of your home directory and everything within it to the /tmp directory, use **cp -a ~ /tmp**

Copying Files

- A special case when working with **cp** are hidden files. By default, hidden files are not copied over. There are three solutions to work hidden files as well:



- **cp /somedir/.* /tmp** This copies all files that have a name starting with a dot (the hidden files, that is) to /tmp. It gives an error message for directories whose name starts with a dot in /somedir, because the **-R** option was not used.
- **cp -a /somedir/ .** This copies the entire directory /somedir, including its contents, to the current directory. So, as a result, a subdirectory somedir will be created in the current directory.
- **cp -a /somedir/. .** This copies all files, regular and hidden, to the current directory.

Moving Files

- To move files, you use the **mv** command. This command removes the file from its current location and puts it in the new location. You can also use it to rename a file (which, in fact, is nothing else than copying and deleting the original file anyway). Let’s take a look at some examples:

- **mv myfile /tmp** Moves the file myfile from the current directory to /tmp.
- **mkdir somefiles; mv somefiles /tmp** This first creates a directory with the name somefiles and then moves this directory to /tmp. Notice that this also works if the directory contains files.
- **mv myfile mynewfile** Renames the file myfile to a new file with the name mynewfile.

Deleting Files

- The last common file administration task is file deletion. To delete files and directories, you use the **rm** command. When used on a single file, the single file is removed. You can also use it on directories that contain files. To do so, include the **-r** option, which again stands for recursive.
- On RHEL 7, the **rm** command prompts for confirmation. If you do not like that, you can use the **-f** option. Make sure that you know what you are doing when using this option, because after using it, there is no way back but the backup tape!

NOTE Many commands have an option that creates recursive behavior. On some commands you use the option **-R**, on other commands you use the option **-r**. That is confusing, but it is just the way it is.

Exercise 3.2 Working with Files

In this exercise, you work with the common file management utilities. Figure 3.1 provides an overview of the directory structure you are working with in this exercise.

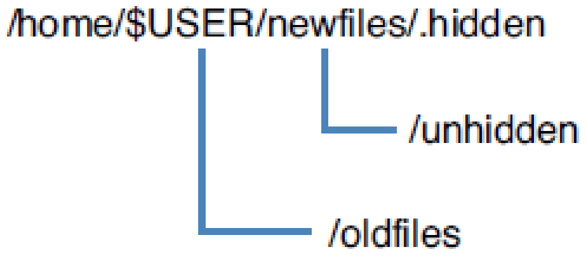


Figure 3.1 Sample directory structure overview.

Exercise 3.2 Working with Files

1. Open a shell as an ordinary user.
2. Type **pwd**. You should be in the directory /home/$USER.
3. Type **mkdir newfiles** and **mkdir oldfiles**. Type **ls**. You'll see the directories you have just created.
4. Type **touch newfiles/.hidden** and **touch newfiles/unhidden**. This creates two files in the directory newfiles.
5. Type **cd oldfiles.**
6. Type **ls -al**. This shows two items only: ., which refers to the current directory; and .., which refers to the item above this (the parent directory).

Exercise 3.2 Working with Files

7. Type **ls -al ../newfiles**. In this command, you are using a relative pathname to refer to the contents of the /home/$USER/newfiles directory.
8. Use the command **cp -a newfiles/**..
9. Type **ls -a**. You see that you have created the subdirectory newfiles into the directory oldfiles.
10. Make sure that you are still in /home/$USER/oldfiles, and type **rm -rf newfiles**.
11. Now use the command **cp -a newfiles/***.. Type **ls -l** to see what has been copied now. You'll see that the hidden file has not been copied.
12. To make sure that you'll copy hidden files as well as regular files, use **cp -a newfiles/**..
13. Verify the command worked this time, using **ls -al**. You'll notice that the hidden as well as the regular files have been successfully copied this time.

Using Links

- Links on Linux are like aliases that are assigned to a file. There are **symbolic links**, and there are **hard links**.
- To understand a link, you need to know a bit about the organization of the Linux file system.

Understanding Hard Links

- Linux stores administrative data about files in inodes. Every file on Linux has an **inode**, and in the inode, important information about the file is stored:
  - The data block where the file contents are stored
  - The creation, access, and modification date
  - Permissions
  - File owners

- Just one important piece of information is not stored in the inode: the name. Names are stored in the directory, and each filename knows which inode it has to address to access further file information. It is interesting to know that an inode does not know which name it has; it just knows how many names are associated with the inode. These names are referred to as hard links.

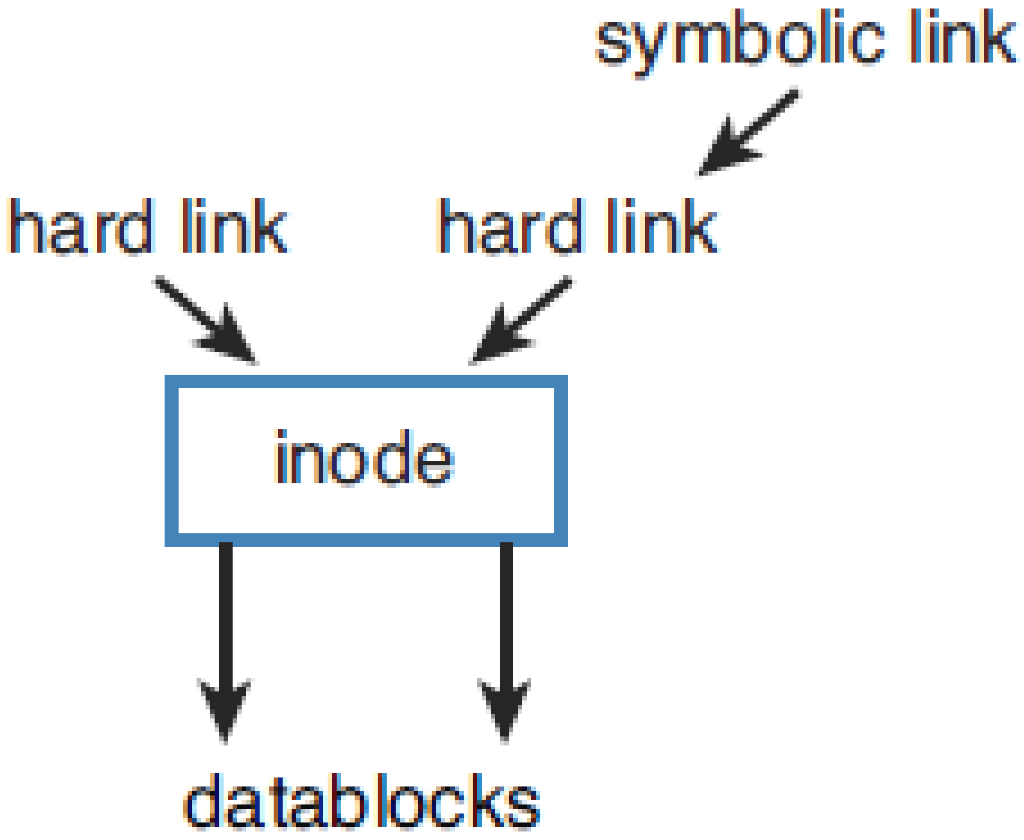
Understanding Hard Links

- When you create a file, you give it a name. Basically, this name is a hard link. On a Linux file system, multiple hard links can be created to a file. This can be useful, because it enables you to access the file from multiple different locations. Some restrictions apply to hard links though:
  - Hard links must exist all on the same device.
  - You cannot create hard links to directories.
  - The number of aliases the original file has. When the last name is removed, the contents are also removed.
- The nice thing about hard links is that no difference exists between the first hard link and the second hard link. They are both just hard links, and if the first hard link that ever existed for a file is removed, that does not impact the other hard links that still exist. The Linux operating system uses links on many locations to make files more accessible.

Understanding Symbolic Links

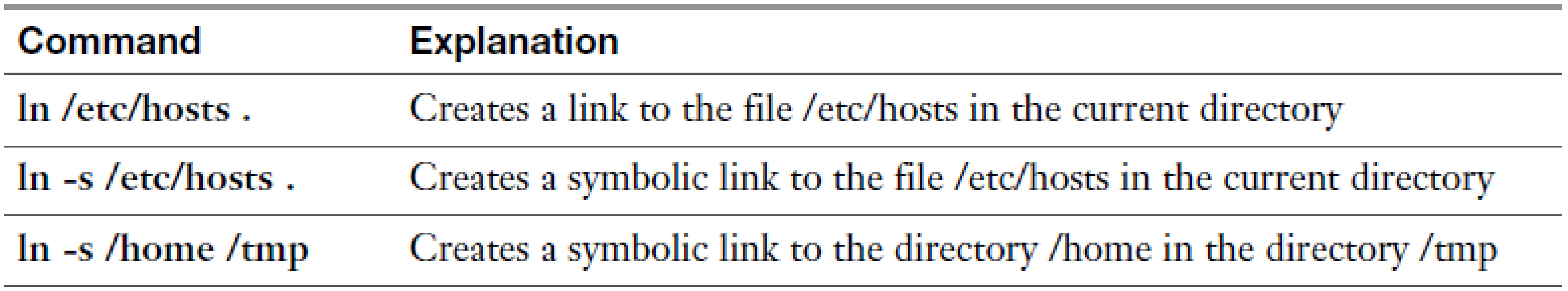
- A symbolic link (also referred to as soft link) does not link directly to the inode but to the name of the file. This makes symbolic links much more flexible, but it also has some disadvantages. The advantage of symbolic links is that they can link to files on other devices, as well as on directories. The major disadvantage is that when the original file is removed, the symbolic link becomes invalid and does not work any longer.

Links and inodes Overview



Creating Links

- Use the **ln** command to create links. It uses the same order of parameters as **cp** and **mv** ; first you mention the source name, followed by the destination name. If you want to create a symbolic link, you use the option **-s** , and then you specify the source and target file or directory.
- One important restriction applies, however; to be able to create hard links, you must be the owner of the item that you want to link to. This is a new security restriction that has been introduced in RHEL 7.

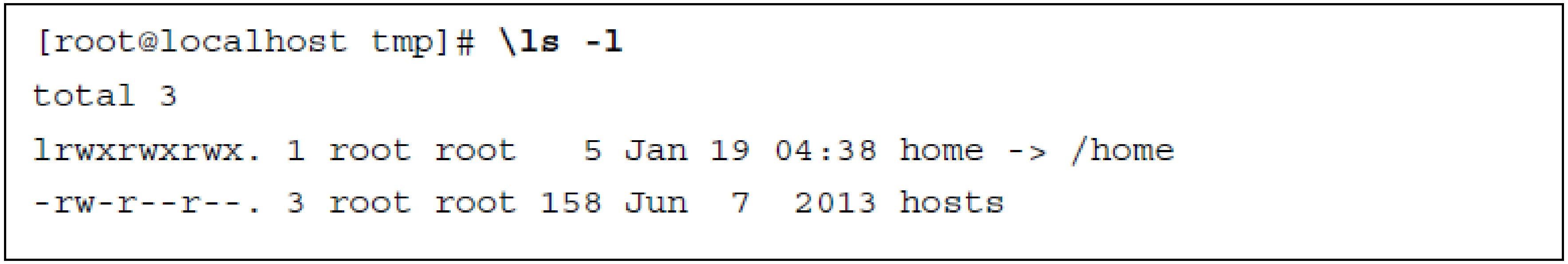


Command	Explanation
ln /etc/hosts .	Creates a link to the file /etc/hosts in the current directory
ln -s /etc/hosts .	Creates a symbolic link to the file /etc/hosts in the current directory
ln -s /home /tmp	Creates a symbolic link to the directory /home in the directory /tmp

Using Links

- The **ls** command will reveal whether a file is a link:
  - In the output of the **ls -l** command, the first character is an l if the file is a symbolic link.
  - If a file is a symbolic link, the output of **ls -l** shows the name of the item it links to after the filename.
  - If a file is a hard link, **ls -l** shows the hard link counter. In the output in Listing 3.4 , this is the number 3 that is right before root root for the hosts file.

Listing 3.4 Showing Link Properties with **ls -l**



```
[root@localhost tmp]# \ls -l
total 3
lrwxrwxrwx. 1 root root   5 Jan 19 04:38 home -> /home
-rw-r--r--. 3 root root 158 Jun  7  2013 hosts
```

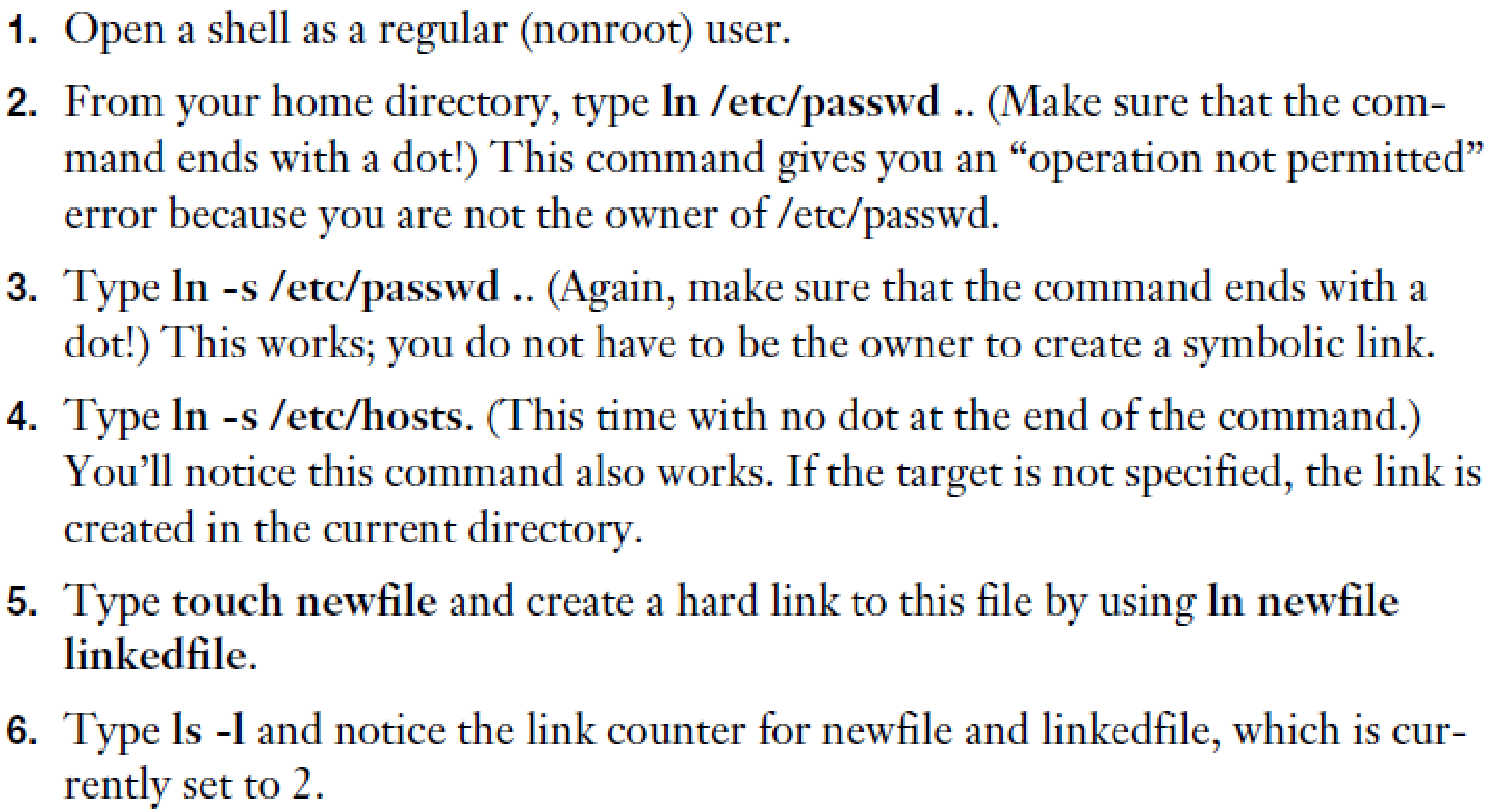
Removing Links

1. Make a directory test in your home directory: **mkdir ~/test**.
2. Copy all files that have a name starting with a, b, c, d, or e from /etc to this directory: **cp /etc/[a-e]* ~/test**.
3. Make sure that you are in your home directory, by using **cd** without arguments.
4. Type **ln -s test link**.
5. Use **rm link**. This removes the link. (Do *not* use **-r** or **-f** to remove links, even if they are subdirectories.)
6. Type **ls -l**. You'll see that the symbolic link has been removed.
7. Let's do it again. Type **ln -s test link** to create the link again.
8. Type **rm -rf link/** (which is what you would get by using bash command line completion)

Removing Links

9. Type **ls**. You'll see that the directory link still exists.
10. Type **ls test/** You'll see the directory test is now empty.

Exercise 3.4 Working with Symbolic Links and Hard Links



1. Open a shell as a regular (nonroot) user.
2. From your home directory, type **ln /etc/passwd .** . (Make sure that the command ends with a dot!) This command gives you an "operation not permitted" error because you are not the owner of /etc/passwd.
3. Type **ln -s /etc/passwd .** . (Again, make sure that the command ends with a dot!) This works; you do not have to be the owner to create a symbolic link.
4. Type **ln -s /etc/hosts**. (This time with no dot at the end of the command.) You'll notice this command also works. If the target is not specified, the link is created in the current directory.
5. Type **touch newfile** and create a hard link to this file by using **ln newfile linkedfile**.
6. Type **ls -l** and notice the link counter for newfile and linkedfile, which is currently set to 2.

Exercise 3.4 Working with Symbolic Links and Hard Links

7. Type **ln -s newfile symlinkfile** to create a symbolic link to newfile.
8. Type **rm newfile.**
9. Type **cat symlinkfile.** You will get a "no such file or directory" error message because the original file could not be found.
10. Type **cat linkedfile.** This gives no problem.
11. Type **ls -l** and look at the way the symlinkfile is displayed. Also look at linkedfile, which now has the link counter set to 1.
12. Type **ln linkedfile newfile.**
13. Type **ls -l** again. You'll see that the original situation has been restored.

Working with Archives and Compressed Files

- Another important file-related task is managing archives and compressed files. To create an archive of files on a Linux computer, the **tar** command is often used. This command was originally designed to stream files to a tape without any compression of the files. If you want to compress files as well, a specific compression tool has to be used, or you need to specify an option that compresses the archive while it is created.
- In this section, you learn how to work with archives and compressed files.

Managing Archives with *tar*

- The **T**ape **AR**chiver (**tar**) utility is used to archive files. Although originally designed to stream files to a backup tape, in its current use tar is used mostly to write files to an archive file. You have to be able to perform three important tasks with tar on the

- RHCSA exam:
  - Create an archive
  - List the contents of an archive
  - Extract an archive

Creating Archives with *tar*

- To create an archive, you use the **tar -cf archivename.tar /files-you-want-toarchive** command. If you want to see what is happening, use the **-v** option as well. To put files in an archive, you need at least read permissions to the file.

- Use **tar-cvf /root/homes.tar /home** as user root to write the contents of the /home directory and everything below it to the file homes.tar in the directory /root.

- Notice the options that are used; the order in these options is important.

Creating Archives with *tar*

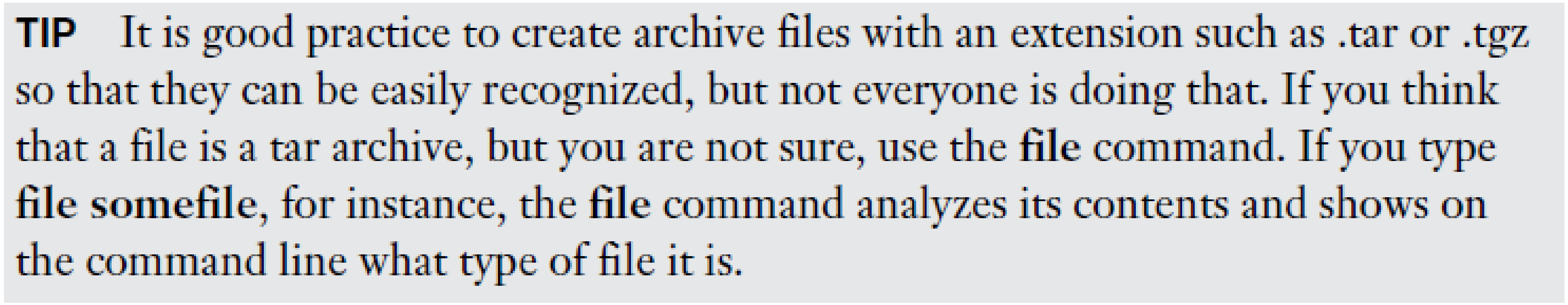
- While managing archives with tar, it is also possible to add a file to an existing archive, or to update an archive. To add a file to an archive, you use the **-r** options.

- Use for instance **tar -rvf /root/homes.tar /etc/hosts** to add the /etc/hosts file to the archive.

- To update a currently existing archive file, you can use the **-u** option. So, use **tar -uvf /root/homes.tar /home** to write newer versions of all files in /home to the archive.

Monitoring Tar Files

- Before extracting a file, it is good to know what might be expected. The option **-t** can be used to find out. Type for instance **tar -tvf /root/homes.tar** to see the contents of the tar archive.



TIP It is good practice to create archive files with an extension such as .tar or .tgz so that they can be easily recognized, but not everyone is doing that. If you think that a file is a tar archive, but you are not sure, use the **file** command. If you type **file somefile**, for instance, the **file** command analyzes its contents and shows on the command line what type of file it is.

Extracting Tar Files

- To extract the contents of an archive, use **tar -cvf /archivename**. This extracts the archive in the *current* directory. That means that if you are in /root when typing **tar -xvf /root/homes.tar** , and the file contains a directory /home, after extracting you'll have a new directory /root/home that contains the entire contents of the file. This might not be what you wanted to accomplish. There are two solutions to put the extracted contents right where you want to have them:
  - Before extracting the archive file, **cd** to the directory where you want to extract the file.
  - Use the option **-C /targetdir** to specify the target directory where you want to extract the file in. If you want to put the contents of the file /root/homes.tar in the directory /tmp, for instance, you can use **tar -xvf homes.tar -C /tmp** .

Extracting Tar Files

- Apart from extracting an entire archive file, it is also possible to extract one file out of the archive. To do so, use **tar -xvf /archivename.tar /file-you-want-to-extract**

- If your archive etc.tar contains the file /etc/hosts that you want to extract, for instance, use
tar -xvf /root/etc.tar /etc/hosts

Using Compression

- Many files contain a lot of redundancy. Compression programs allow you to make files take less disk space by taking out that redundancy. In all examples of the **tar** command that you have seen so far, not a single byte has been compressed. Originally, after creating the archive, it had to be compressed with a separate compression utility, such as **gzip** or **bzip2** .

- After having created home.tar, you can compress it with **gzip home.tar**

- **gzip** replaces home.tar with its compressed version, **home.tar.gz**, which takes significantly less space.

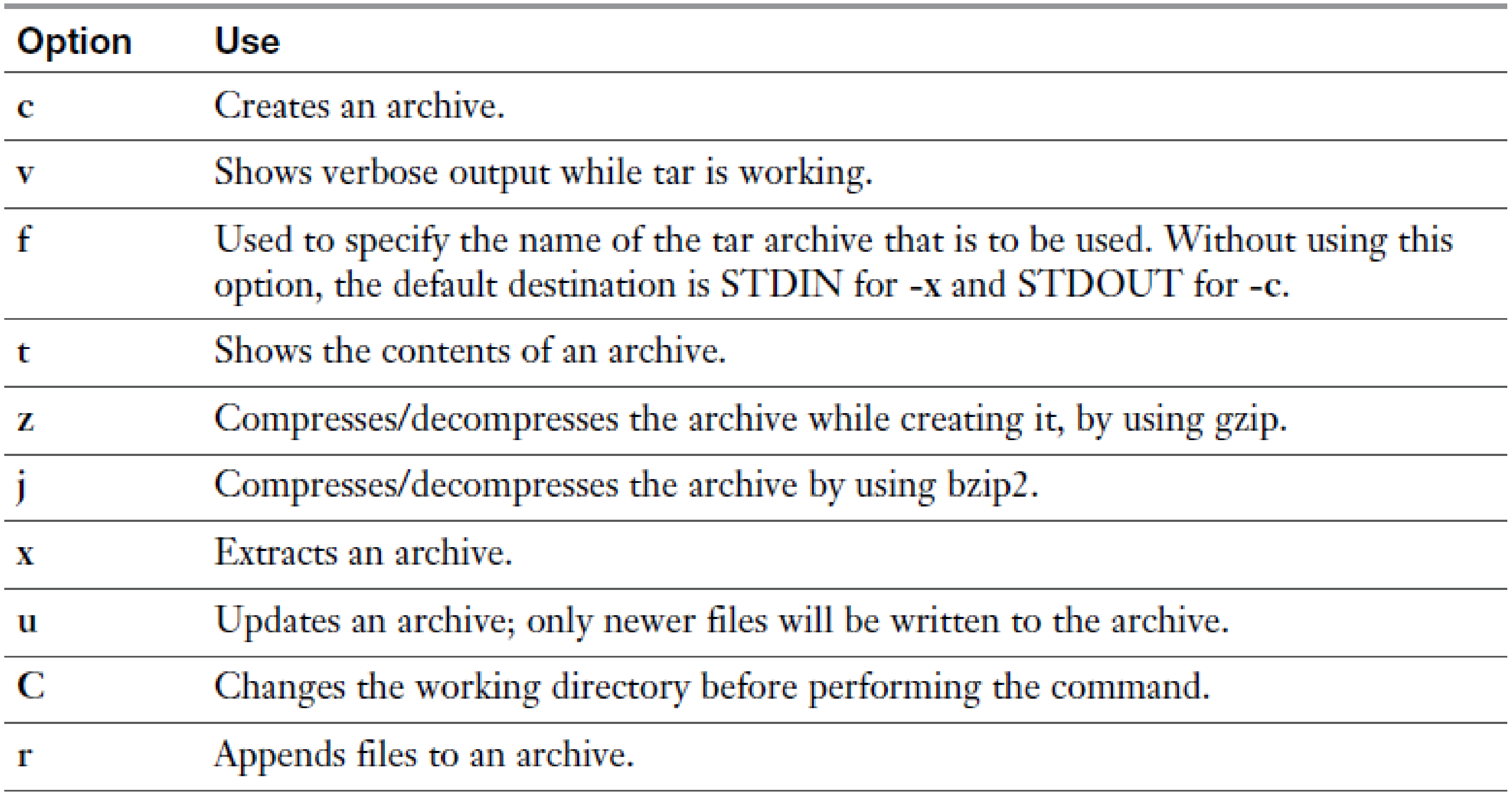
Using Compression

- As an alternative to using **gzip** , you can use the **bzip2** utility. Originally, bzip2 used a more efficient encryption algorithm, which resulted in smaller file sizes, but currently it hardly makes a difference anymore with the result of the **gzip** utility.

- To decompress files that have been compressed with **gzip** or **bzip2** , you can use the **gunzip** and **bunzip2** utilities.

- As an alternative to using these utilities from the command line, you can include the **-z** (*gzip*) or **-j** (*bzip2*) options while creating the archive with tar.

Overview of *tar* Options



Option	Use
c	Creates an archive.
v	Shows verbose output while tar is working.
f	Used to specify the name of the tar archive that is to be used. Without using this option, the default destination is STDIN for **-x** and STDOUT for **-c**.
t	Shows the contents of an archive.
z	Compresses/decompresses the archive while creating it, by using gzip.
j	Compresses/decompresses the archive by using bzip2.
x	Extracts an archive.
u	Updates an archive; only newer files will be written to the archive.
C	Changes the working directory before performing the command.
r	Appends files to an archive.

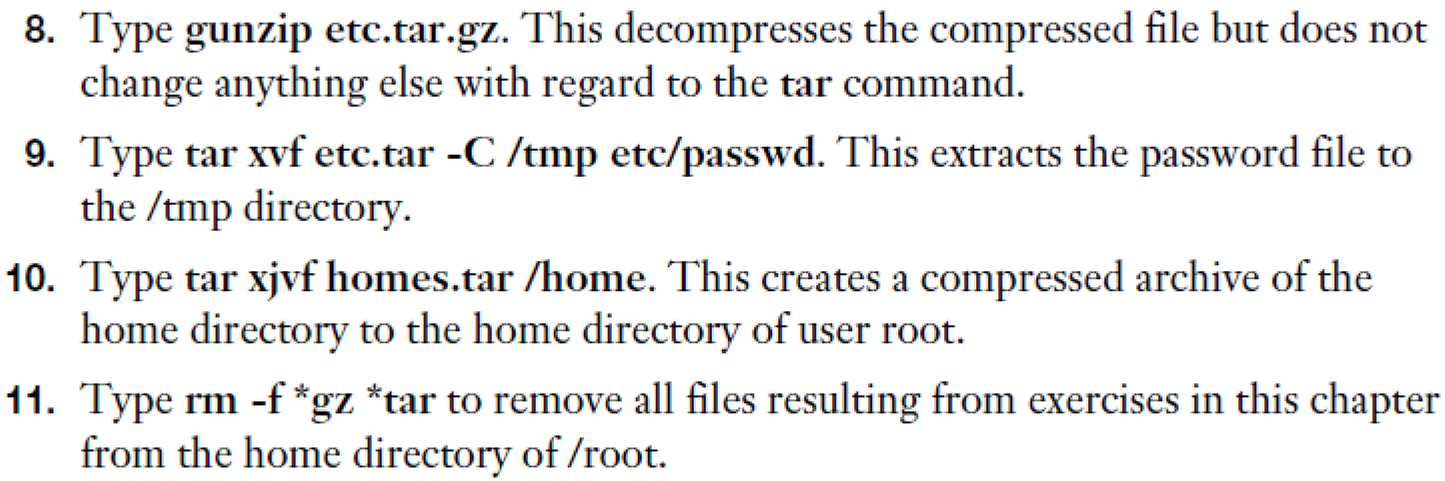
Exercise 3.5 Using *tar*

1. Open a root shell on your server. By logging in, the home directory of user root will become the current directory, so all relative filenames used in this exercise refer to /root/.
2. Type **tar -cvf etc.tar /etc** to archive the contents of the /etc directory.
3. Type **file etc.tar** and read the information that is provided by the command. This should look like the following:

```
[root@server1 ~]# file etc.tar
etc.tar: POSIX tar archive (GNU)
```

4. Type **gzip etc.tar**.
5. Type **tar tvf etc.tar.gz**. Notice that the **tar** command has no issues reading from a gzip compressed file. Also notice that the archive content consists of all relative filenames.
6. Type **tar xvf etc.tar.gz etc/hosts**.
7. Type **ls -R**. Notice that a subdirectory etc has been created in the current directory. In this subdirectory, the file hosts has been restored.

Exercise 3.5 Using *tar*



8. Type **gunzip etc.tar.gz**. This decompresses the compressed file but does not change anything else with regard to the **tar** command.
9. Type **tar xvf etc.tar -C /tmp etc/passwd**. This extracts the password file to the /tmp directory.
10. Type **tar xjvf homes.tar /home**. This creates a compressed archive of the home directory to the home directory of user root.
11. Type **rm -f *gz *tar** to remove all files resulting from exercises in this chapter from the home directory of /root.

Summary

- In this chapter, you learned how to work with essential file management tools. You learned how the Linux directory structure is organized by default, and you learned what file types to expect in which directories. You also learned how to find your way in the directory structure and to work with files.

Define Key Terms

- Define the following key terms:
  - FSH
  - Directory
  - Mount
  - Device
  - Folder
  - Root directory
  - Path
  - Hard link
  - Symbolic link
  - Absolute filename
  - Relative filename
  - Inode
  - Tar
  - Gzip
  - Compression

Lab 3.1

1. Log in as user root. In the home directory of root, create one archive file that contains the contents of the /home and the /etc directory. Use the name /root/essentials.tar for the archive file.
2. Copy this archive to the /tmp directory. Also create a hard link to this file in the / directory.
3. Rename the file /essentials.tar to /archive.tar.
4. Create a symbolic link in the home directory of the user root that refers to /archive.tar. Use the name link.tar for the symbolic link.
5. Remove the file /archive.tar and see what happened to the symbolic link. Remove the symbolic link also.
6. Compress the /root/essentials.tar file.

Chapter 4:

Working with Text Files

Chapter 4 Objectives

- The following topics are covered in this chapter:
  - Using Common Text File-Related Tools
  - A Primer to Using Regular Expressions
  - Using grep to Analyze Text
  - Working with Other Useful Text Processing Utilities

- The following RHCSA exam objectives are covered in this chapter:
  - Using grep and regular expressions to analyze text
  - Create and edit text files

Using Common Text File-Related Tools

Table 4.2 Essential Tools for Managing Text File Contents

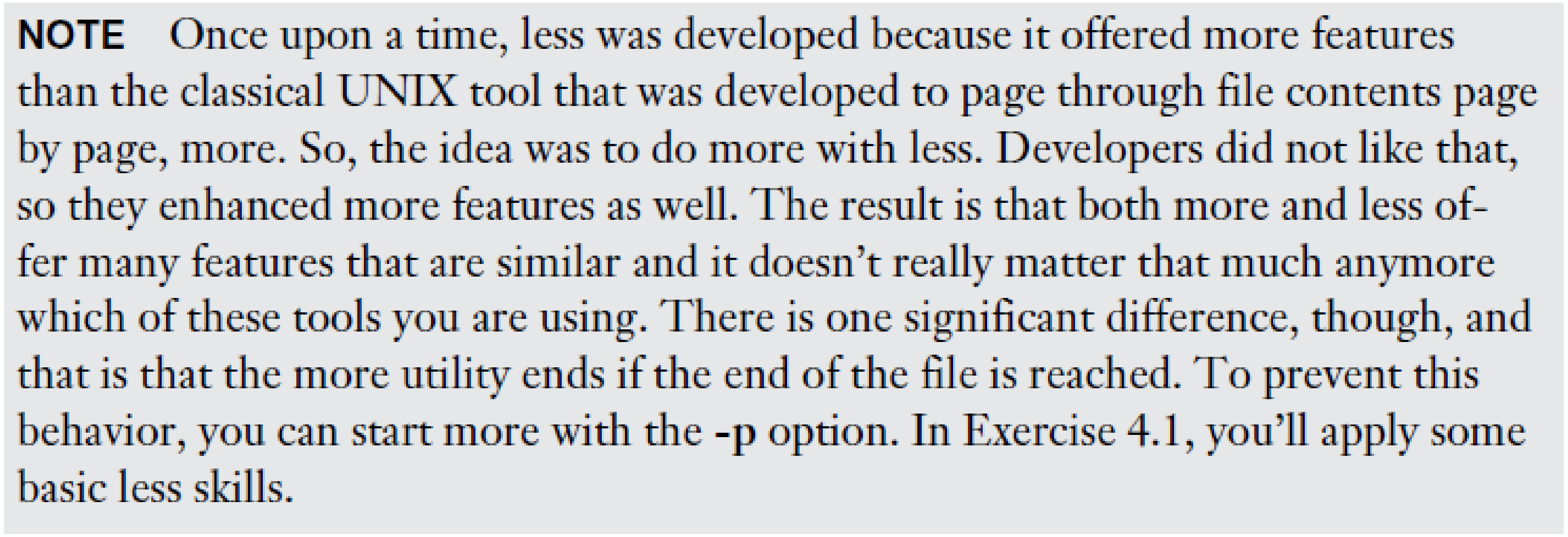
Command	Explanation
less	Opens the text file in a pager, which allows for easy reading of the text file
cat	Dumps the contents of the text file on the screen
head	Shows the first 10 lines of the text file
tail	Shows the last 10 lines of the text file
cut	Used to filter specific columns or characters from a text file
sort	Sorts contents of a text file
wc	Counts the number of lines, words, and characters in a file

Apart from using these commands on a text file, they may also prove very useful when used in pipes. You can use the command **less /etc/passwd**, for example, to open the contents of the /etc/passwd file in the **less** pager, but you can also use the command **ps aux | less**, which sends the output of the command **ps aux** to the pager less to allow for easy reading.

Doing *More* with *Less*

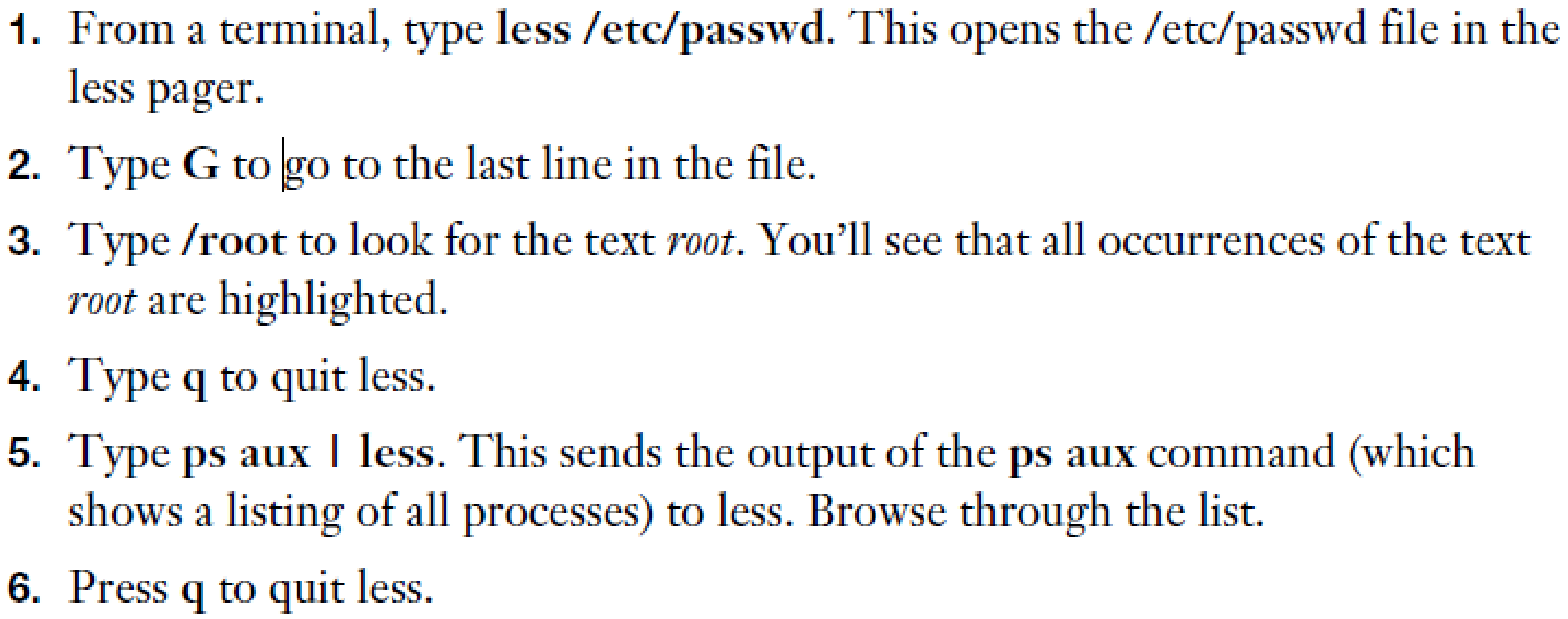
- In many cases, as a Linux administrator you'll need to read the contents of text files. The less utility offers a convenient way to do so. To open the contents of a text file in less, just type **less** followed by the name of the file you want to see, as in **less /etc/passwd**
- From less, you can use the **PageUp** and **PageDown** keys on your keyboard to browse through the file contents. Seen enough? Then you can press **q** to quit less.
- Also very useful is that you can easily search for specific contents in less using **/sometext** for a forward search and **?sometext** for a backward search.
  Repeat the last search by using **n**.

Doing *More* with *Less*



NOTE Once upon a time, less was developed because it offered more features than the classical UNIX tool that was developed to page through file contents page by page, more. So, the idea was to do more with less. Developers did not like that, so they enhanced more features as well. The result is that both more and less offer many features that are similar and it doesn't really matter that much anymore which of these tools you are using. There is one significant difference, though, and that is that the more utility ends if the end of the file is reached. To prevent this behavior, you can start more with the **-p** option. In Exercise 4.1, you'll apply some basic less skills.

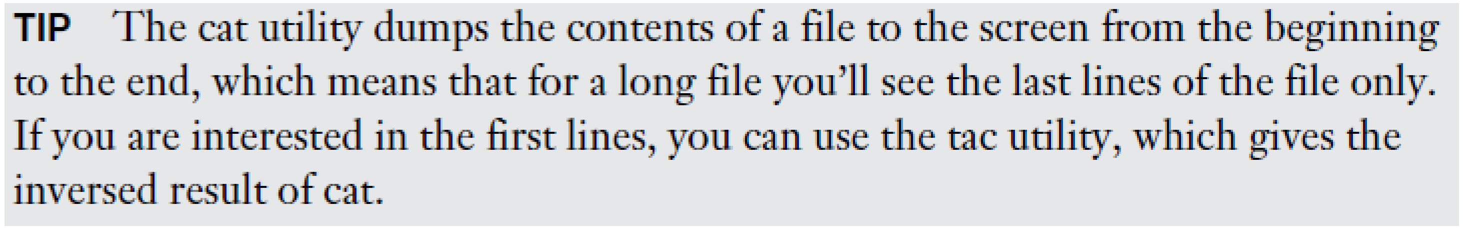
Exercise 4.1 Applying Basic Less Skills



1. From a terminal, type **less /etc/passwd**. This opens the /etc/passwd file in the less pager.
2. Type **G** to go to the last line in the file.
3. Type **/root** to look for the text *root*. You'll see that all occurrences of the text *root* are highlighted.
4. Type **q** to quit less.
5. Type **ps aux | less**. This sends the output of the **ps aux** command (which shows a listing of all processes) to less. Browse through the list.
6. Press **q** to quit less.

Showing File Contents with *cat*

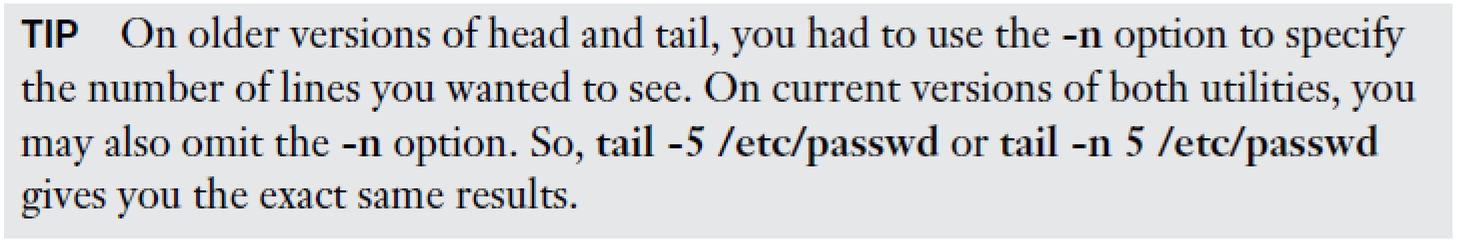
- The less utility is useful to read long text files. If a text file is not that long, you are probably better off using cat. This tool just dumps the contents of the text file on the terminal it was started from. This is convenient is the text file is short. If the text file is long, however, you'll see all contents scrolling by on the screen, and only the lines that fit on the terminal screen are displayed. Using cat is simple.
- Just type **cat** followed by the name of the file you want to see. For instance, use **cat /etc/passwd** to show the contents of this file on your computer screen.



TIP The cat utility dumps the contents of a file to the screen from the beginning to the end, which means that for a long file you'll see the last lines of the file only. If you are interested in the first lines, you can use the tac utility, which gives the inversed result of cat.

First or Last Lines of a File with *head* and *tail*

- Using head on a text file will show by default the first 10 lines of that file. Using tail on a text file shows the last 10 lines by default. You can adjust the number of lines that are shown by adding **-n** followed by the number you want to see.
- So, **tail -n 5 /etc/passwd** shows the last five lines of the /etc/passwd file.



TIP On older versions of head and tail, you had to use the **-n** option to specify the number of lines you wanted to see. On current versions of both utilities, you may also omit the **-n** option. So, **tail -5 /etc/passwd** or **tail -n 5 /etc/passwd** gives you the exact same results.

First or Last Lines of a File with *head* and *tail*

- Another useful option that you can use with tail is **-f** . This option starts by showing you the last 10 lines of the file you’ve specified, but it refreshes the display as new lines are added to the file. This is convenient for monitoring log files.
- The command **tail -f /var/log/messages** is a common command to show in real-time messages that are written to the main log file /var/log/messages.
- Suppose, for instance, that you want to see line number 11 of the /etc/passwd file. To do that, use
  head -n 11 /etc/passwd | tail -n 1

Exercise 4.2 Basic *head* and *tail* Operations

1. Type **tail -f /var/log/messages** . You'll see the last lines of /var/log/messages being displayed. The file doesn't close automatically.
2. Type **Ctrl+C** to quit the previous command.
3. Type **head -n 5 /etc/passwd** to show the first five lines in /etc/passwd.
4. Type **tail -n 2 /etc/passwd** to show the last two lines of /etc/passwd.
5. Type **head -n 5 /etc/passwd | tail -n 1** to show only line number 5 of the /etc/passwd file.

Filtering Specific Columns with *cut*

- When working with text files, it can be useful to filter out specific fields. Imagine that you need to see a list of all users in the /etc/passwd file. In this file, several fields are defined, of which the first contains the name of the users who are defined.
- To filter out a specific field, the **cut** command is useful. To do this, use the **-d** option to specify the field delimiter followed by **-f** with the number of the specific field you want to filter out.
- So, the complete command is **cut -d : -f 1 /etc/passwd** if you want to filter out the first field of the /etc/passwd file.

Filtering Specific Columns with *cut*

Listing 4.1 Filtering Specific Fields with **cut**

```
[root@localhost ~]# cut -f 1 -d : /etc/passwd
root
bin
daemon
adm
lp
sync
shutdown
halt
...
```

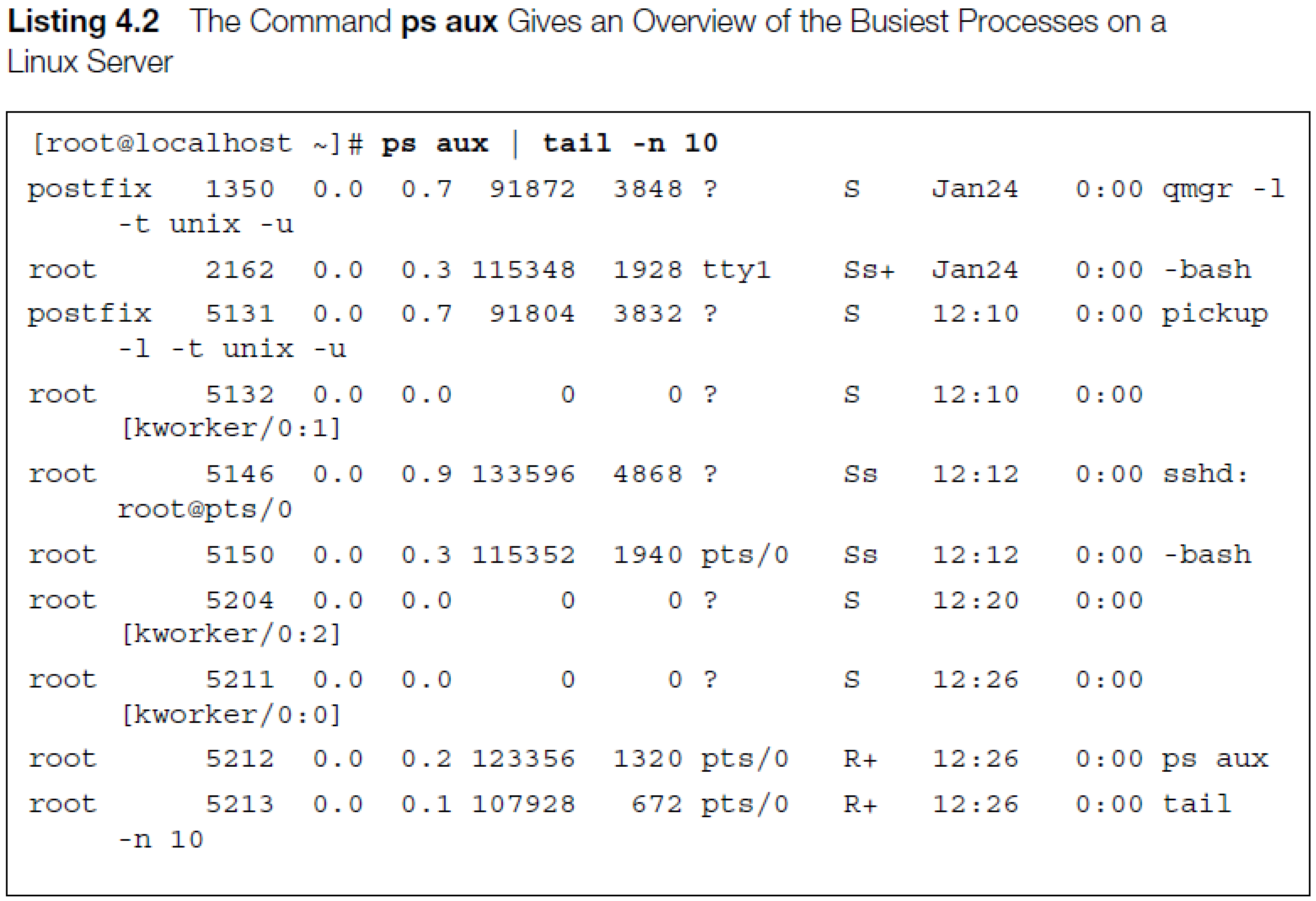
Sorting File Contents and Output with *sort*

- Another very useful command to use on text file is **sort** .
- As you can probably guess, this command sorts text.
- If you type **sort /etc/passwd** , for instance, the content of the /etc/passwd file is sorted in alphabetic order.
- You can use the **sort** command on the output of a command also, as in **cut -f 1 -d : /etc/passwd | sort** , which sorts the contents of the first column in the /etc/passwd file.
- By default, the **sort** command sorts in alphabetic order. In some cases, that is not convenient because the content that needs sorting may be numeric or in another format. The **sort** command offers different options to help sorting these specific types of data.

Sorting File Contents and Output with *sort*

- Type, for instance, **cut -f 2 -d : /etc/passwd | sort -n** to sort the second field of the /etc/passwd file in numeric order. It can be useful also to sort in reverse order; if you use the command **du -h | sort -rn** , you get a list of files sorted with the biggest file in that directory listed first.
- You can also use the **sort** command and specify which column you want to sort.
- To do this, use **sort -k3 -t : /etc/passwd** , for instance, which uses the field separator **:** to sort the third column of the /etc/passwd file.

Sorting File Contents and Output with *sort*



Listing 4.2 The Command **ps aux** Gives an Overview of the Busiest Processes on a Linux Server

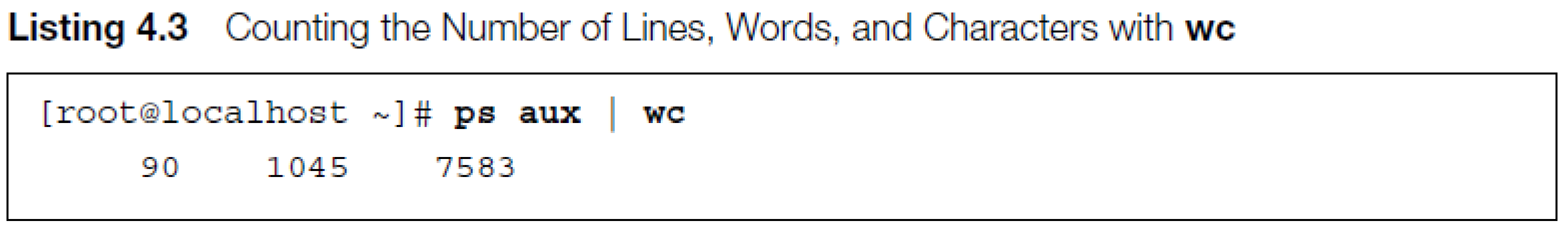
```
[root@localhost ~]# ps aux | tail -n 10
postfix   1350  0.0  0.7  91872  3848 ?        S    Jan24   0:00 qmgr -l
     -t unix -u
root      2162  0.0  0.3 115348  1928 tty1     Ss+  Jan24   0:00 -bash
postfix   5131  0.0  0.7  91804  3832 ?        S    12:10   0:00 pickup
     -l -t unix -u
root      5132  0.0  0.0      0     0 ?        S    12:10   0:00
     [kworker/0:1]
root      5146  0.0  0.9 133596  4868 ?        Ss   12:12   0:00 sshd:
     root@pts/0
root      5150  0.0  0.3 115352  1940 pts/0    Ss   12:12   0:00 -bash
root      5204  0.0  0.0      0     0 ?        S    12:20   0:00
     [kworker/0:2]
root      5211  0.0  0.0      0     0 ?        S    12:26   0:00
     [kworker/0:0]
root      5212  0.0  0.2 123356  1320 pts/0    R+   12:26   0:00 ps aux
root      5213  0.0  0.1 107928   672 pts/0    R+   12:26   0:00 tail
     -n 10
```

To sort the output of this command directly on the third column, use the command **ps aux | sort -k3**

Counting Lines, Words, and Characters with *wc*

- When working with text files, you sometimes get a large amount of output. Before deciding which approach works best in a specific case, you might want to have an idea about the amount of text you are dealing with. In that case, the **wc** command is useful. In its output, this command gives three different results: the number of lines, the number of words, and the number of characters.

- Consider, for example, the **ps aux** command. When executed as root, this command gives a list of all processes running on a server. One solution to count how many processes there are exactly is to pipe the output of **ps aux** through **wc** , as in **ps aux | wc**



Listing 4.3 Counting the Number of Lines, Words, and Characters with **wc**

```
[root@localhost ~]# ps aux | wc
     90     1045     7583
```

A Primer to Using Regular Expressions

```
[root@localhost ~]# tail -n 6 /etc/passwd
anna:x:1000:1000::/home/anna:/bin/bash
rihanna:x:1001:1001::/home/rihanna:/bin/bash
annabel:x:1002:1002::/home/annabel:/bin/bash
anand:x:1003:1003::/home/anand:/bin/bash
joanna:x:1004:1004::/home/joanna:/bin/bash
joana:x:1005:1005::/home/joana:/bin/bash
```

- Now suppose that you are looking for the user anna. In that case, you could use the general regular expression parser grep to look for that specific string in the file /etc/passwd by using the command **grep anna /etc/passwd**

A Primer to Using Regular Expressions

```
[root@localhost ~]# grep anna /etc/passwd
anna :x:1000:1000::/home/ anna :/bin/bash
rihanna :x:1001:1001::/home/rih anna :/bin/bash
annabel:x:1002:1002::/home/ anna bel:/bin/bash
joanna :x:1004:1004::/home/jo anna :/bin/bash
```

A regular expression is a search pattern that allows you to look for specific text in an advanced and flexible way.

Using Line Anchors

- To show only lines that start with the text you are looking for, you can use the regular expression ^ (in this case, to indicate that you are looking only for lines where *anna* is at the beginning of the line

```
[root@localhost ~]# grep ^anna /etc/passwd
anna :x:1000:1000::/home/anna:/bin/bash
annabel:x:1002:1002::/home/annabel:/bin/bash
```

- Another regular expression that relates to the position of specific text in a specific line is $, which states that the line ends with some text. For instance, the command **grep ash$ /etc/passwd** shows all lines in the /etc/passwd file that end with the text *ash.*

Using Escaping in Regular Expressions

- Although not mandatory, when using regular expressions it is a good idea to use escaping to prevent the regular expression from being interpreted by the shell. In many cases, it is not really necessary to use escaping; in some cases, the regular expression fails without escaping. To prevent this from ever happening, it is a good idea to put the regular expression between quotes.
- So, instead of typing **grep ^anna /etc/passwd** ,
  it is better use **grep '^anna' /etc/passwd,**
  even if in this case both examples work.

Using Wildcards and Multipliers

- In some cases, you might know which text you are looking for, but you might not know how the specific text is written. Or you just want to use one regular expression to match different patterns. In those cases, wildcards and multipliers come in handy.
- To start with, there is the **.** regular expression. This is used as a wildcard character to look for one specific character. So, the regular expression **r.t** would match the strings rat, rot, and rut.
- In some cases, you might want to be more specific about the characters you are looking for. If that is the case, you can specify a range of characters that you are looking for. For instance, the regular expression **r[aou]t** matches the strings rat, rut, as well as rot.

Using Wildcards and Multipliers

- Another useful regular expression is the multiplier *****. This matches zero or more of the previous character. That does not seem to be very useful, but indeed it is, as you will see in the examples at the end of this section.
- If you know exactly how many of the previous character you are looking for, you can specify a number also, as in **re\{2\}d**, which would match *red* as well as *reed*.
- The last regular expression that is useful to know about is **?**, which matches zero or one of the previous character.
- Table 4.3 provides an overview of the most important regular expressions.

Using Wildcards and Multipliers

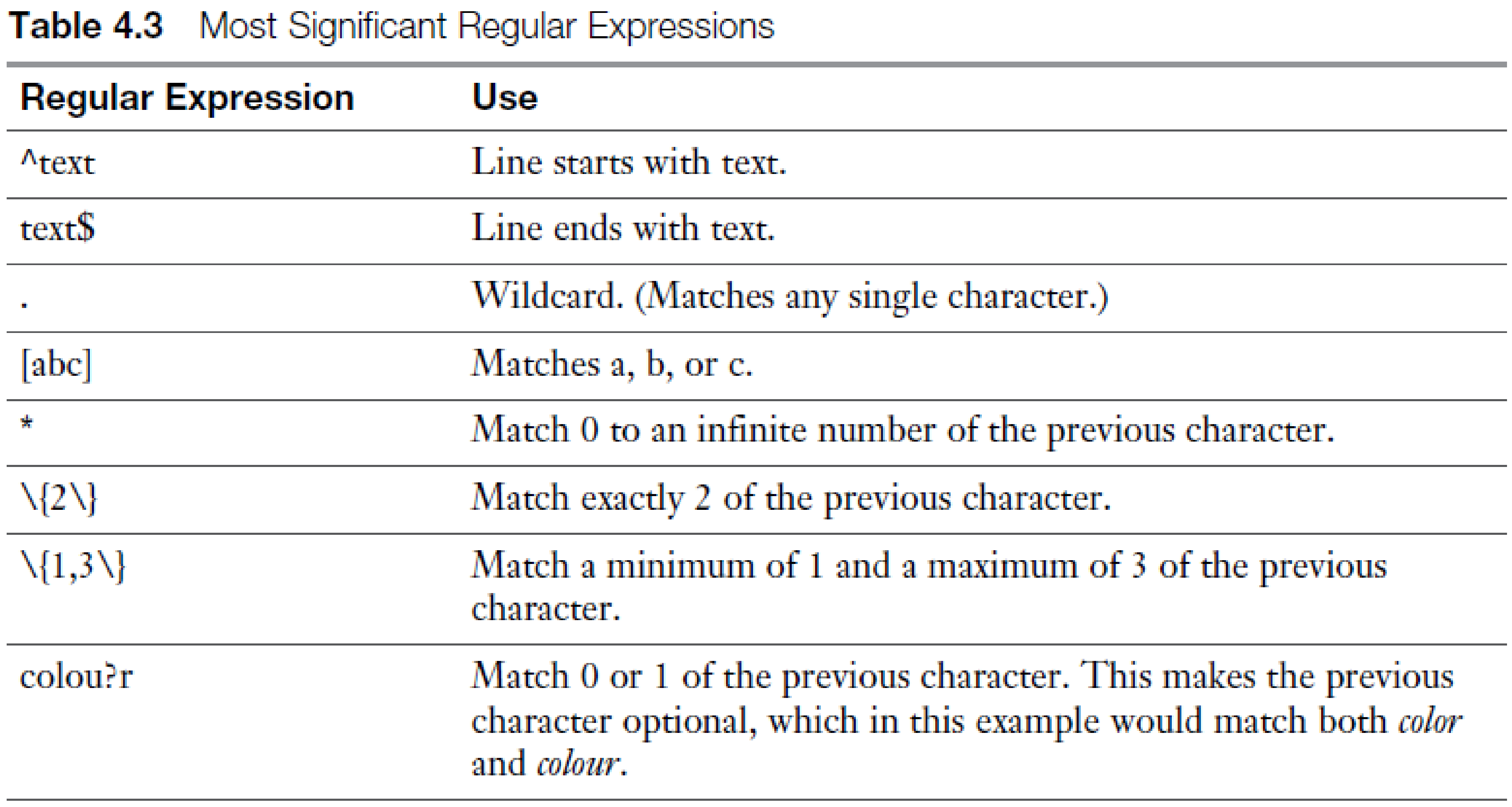
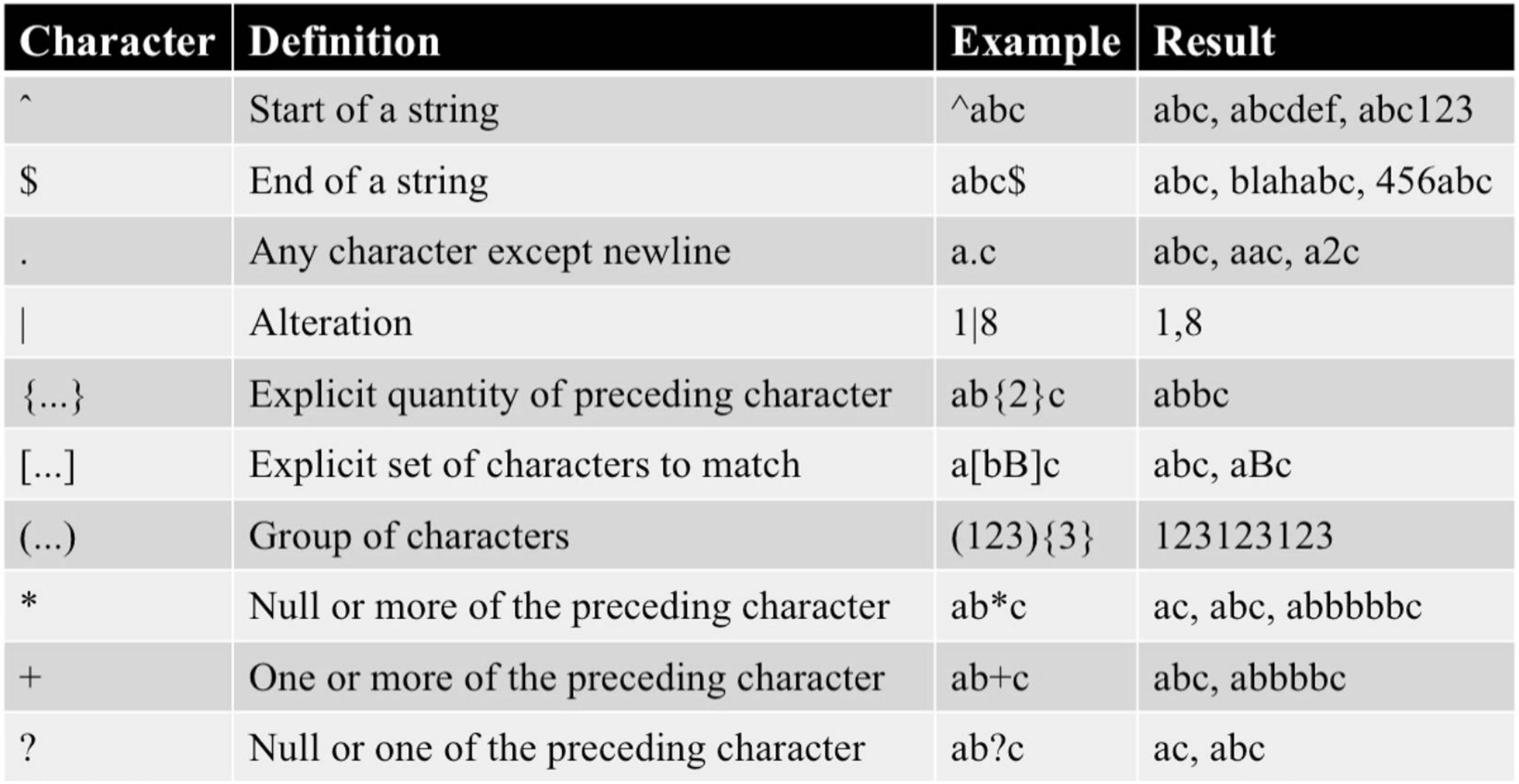


Table 4.3 Most Significant Regular Expressions

Regular Expression	**Use**
^text	Line starts with text.
text$	Line ends with text.
.	Wildcard. (Matches any single character.)
[abc]	Matches a, b, or c.
*	Match 0 to an infinite number of the previous character.
\{2\}	Match exactly 2 of the previous character.
\{1,3\}	Match a minimum of 1 and a maximum of 3 of the previous character.
colou?r	Match 0 or 1 of the previous character. This makes the previous character optional, which in this example would match both *color* and *colour*.

```
man 7 glob
man 7 regex
```

Using Wildcards and Multipliers



Character	Definition	Example	Result
^	Start of a string	^abc	abc, abcdef, abc123
$	End of a string	abc$	abc, blahabc, 456abc
.	Any character except newline	a.c	abc, aac, a2c
|	Alteration	1|8	1,8
{...}	Explicit quantity of preceding character	ab{2}c	abbc
[...]	Explicit set of characters to match	a[bB]c	abc, aBc
(...)	Group of characters	(123){3}	123123123
*	Null or more of the preceding character	ab*c	ac, abc, abbbbbc
+	One or more of the preceding character	ab+c	abc, abbbbc
?	Null or one of the preceding character	ab?c	ac, abc

Using Wildcards and Multipliers

```
"/web(/.*)?"
```

- In this regular expression, the text */web* is referred to. This text string can be followed by the regular expression (/.*)?, which means zero or one (/.*), which in fact means that it can be followed by nothing or (/.*). The (/.*) refers to a slash which may be followed by an unlimited number of characters. To state it differently, the regular expression refers to the text */web* which may or may not be followed by any characters

Using Wildcards and Multipliers

```
ls host*
ls *host*
ls ?ost
ls [hm]ost
ls [!hm]ost
ls [0-9][0-9]ost
```

Using *grep* to Analyze Text

- The ultimate utility to work with regular expressions is grep, which stands for “general regular expression parser.”
- Quite a few examples that you have seen already were based on the **grep** command. The **grep** command has a couple of useful options to make it even more efficient.

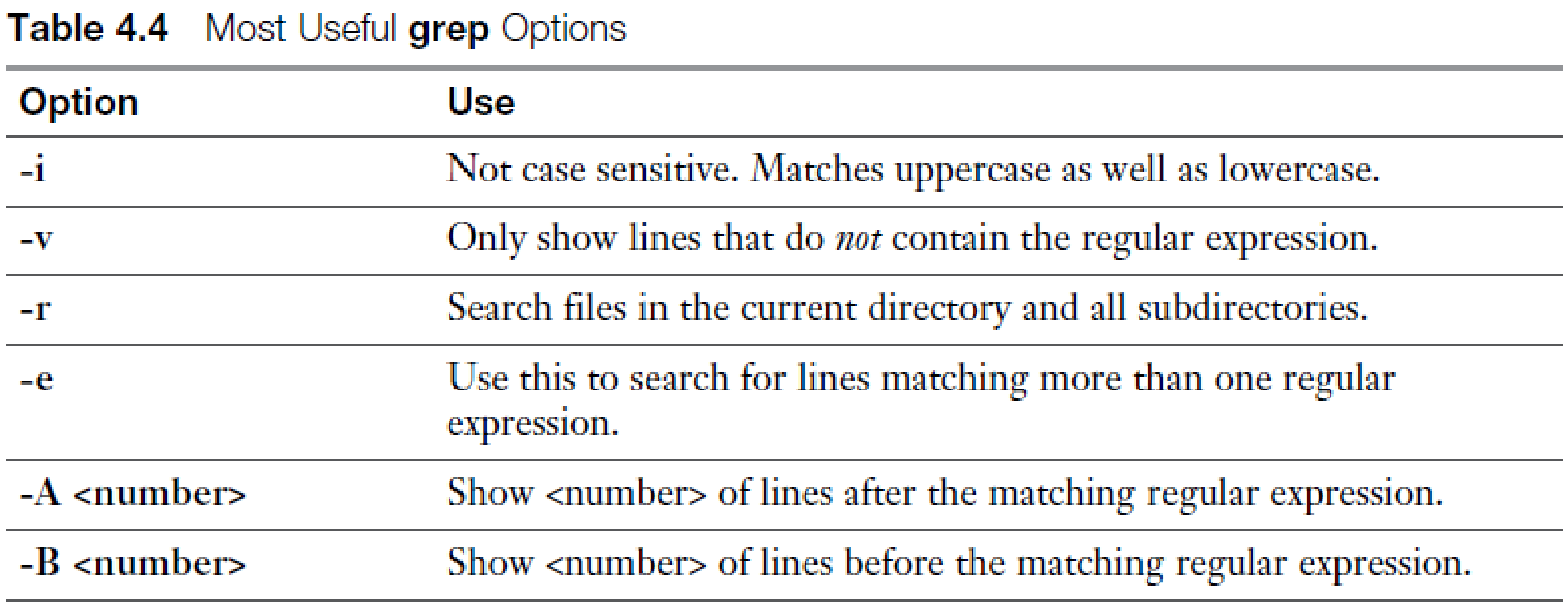


Table 4.4 Most Useful **grep** Options

Option	Use
-i	Not case sensitive. Matches uppercase as well as lowercase.
-v	Only show lines that do *not* contain the regular expression.
-r	Search files in the current directory and all subdirectories.
-e	Use this to search for lines matching more than one regular expression.
-A <number>	Show <number> of lines after the matching regular expression.
-B <number>	Show <number> of lines before the matching regular expression.

Exercise 4.3 Using Common grep Options

1. Type **grep '^#' /etc/sysconfig/sshd**. This shows that the file /etc/sysconfig/sshd contains a number of lines that start with the comment sign #.
2. To view the configuration lines that really matter, type **grep -v '^#' /etc/sysconfig/sshd**. This shows only lines that do not start with a #.
3. Now type **grep -v '^#' /etc/sysconfig/sshd -B 5**. This shows lines that are not starting with a # sign but also the five lines that are directly before that line, which is useful because in these lines you'll typically find comments on how to use the specific parameters. However, you'll also see that many blank lines are displayed.
4. Type **grep -v -e '^#' -e '^$' /etc/sysconfig/sshd**. This excludes all blank lines and lines that start with #.

Other Useful Text Processing Utilities

- The grep utility is a powerful utility that allows you to work with regular expressions. It is not the only utility, though. Some even more powerful utilities exist, like **awk** and **sed**. Both utilities are extremely rich and merit a book by themselves. As a Linux administrator in the twenty-first century, you do not have to be a specialist in using these utilities anymore. It does make sense, however, to know how to perform some common tasks using these utilities. The most useful use cases are summarized in the following examples:

Other Useful Text Processing Utilities

```
awk -F : '{ print $4 }' /etc/passwd
```

- This command shows the fourth line from /etc/passwd. This is something that can be done by using the cut utility as well, but the awk utility is more successful in distinguishing the fields that are used in command output of files.

```
awk -F : '/user/ { print $4 }' /etc/passwd
```

- This command searches the /etc/passwd file for the text user and will print the fourth field of any matching line.

Other Useful Text Processing Utilities

```
sed -n 5p /etc/passwd
```

- In this example, the “stream editor” sed is used to print the fifth line from the /etc/passwd file.

```
sed -i s/old-text/new-text/g ~/myfile
```

- In this example the sed utility is used to search the text *old-text* in ~/myfile and on all occurrences replace it with the text *new-text*. Notice that the default sed behavior is to write the output to STDOUT, but the option **-i** will write the result directly to the file.

Other Useful Text Processing Utilities

```
sed -i -e '2d' ~/myfile
```

- You will like the preceding example if you've ever had a utility containing a specific line in a file that was erroneous. With this command, you can delete a line based on a specific line number. You can also make more complicated references to line numbers.

- Use, for instance, **sed -i -e '2d;20,25d' ~/myfile** to delete lines 2 and 20 through 25 in the file ~/myfile

TIP Do not focus on awk and sed too much. These are amazing utilities, but many of the things that can be accomplished using awk and sed can be done using other tools as well. The awk and sed tools are very rich, and you can easily get lost in them if you are trying to dig too deep.

Summary

- In this chapter, you learned how to work with text files. You acquired some important skills like searching text files with grep and displaying text files or part of them with different utilities. You have also learned how regular expressions can be used to make the search results more specific and learned about the very sophisticated utilities awk and sed, which allow you to perform more advanced operations on text files.

Define Key Terms

- Define the following key terms:
  - Regular expression
  - Pager
  - Escaping
  - Wildcards
  - Multipliers
  - Line anchors

Lab 4.1

1. Describe two ways to show line 5 from the /etc/passwd file?
2. How would you locate all text files on your server that contain the current IP address? Do you need a regular expression to do this?
3. You have just used the **sed** command that replaces all occurrences of the text *Administrator* with *root* . Your Windows administrators do not like that very much. How do you revert?
4. Assuming that in the **ps aux** command the fifth line contains information about memory utilization, how would you process the output of that command to show the process that has the most heavy memory utilization first in the results list?
5. Which command enables you to filter the sixth column of **ps aux** output?
6. How do you delete the sixth line from the file ~/myfile?

Chapter 5:

Connecting to a RHEL Server

Chapter 5 Objectives

- The following topics are covered in this chapter:
  - Working on Local Consoles
  - Using SSH and Related Utilities

- The following RHCSA exam objectives are covered in this chapter:
  - Access remote systems using SSH
  - Switch users in multiple targets
  - Boot, reboot, and shut down a system normally
  - Securely transfer files between systems
  - Configure key-based authentication for SSH

Working on Local Consoles

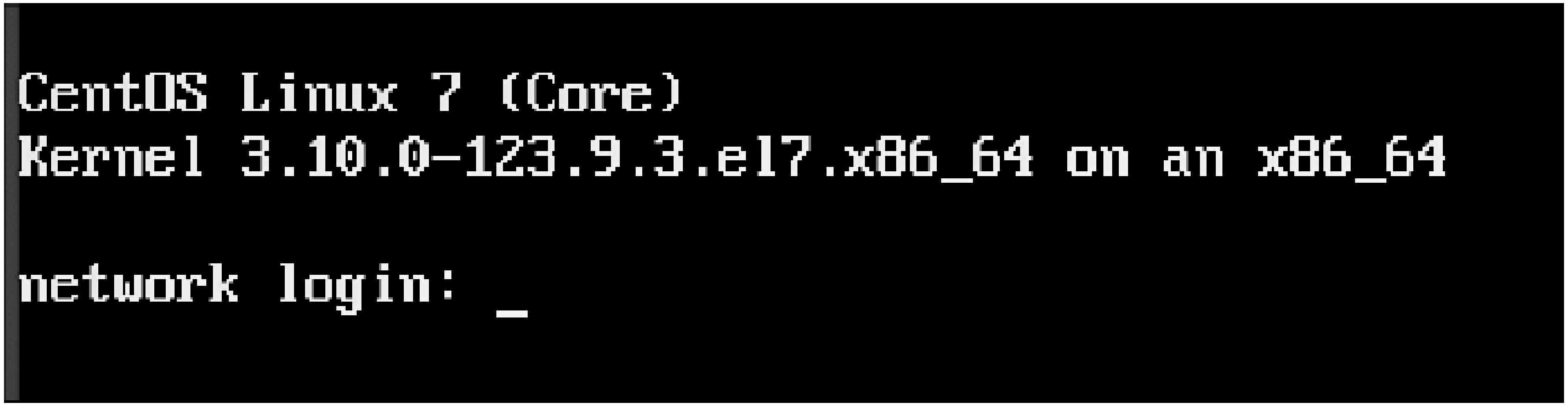
- You have already learned how to log in on Linux by using a graphical console. In this section, you learn some more about the possibilities you have while working from either a graphical Linux console or a text-based Linux console.

- A terminal is an environment that is opened on the console, and which provides access to a text shell, which is the command-line environment that can be used to type commands. A terminal can be offered through a window while using a graphical console, but it can also be opened as the only thing that you see in a textual console. This means that on a textual environment, the words console and terminal are more or less equivalent. In a graphical environment, they are not. Think of it like this: You can have multiple terminals open on a console, but you cannot have multiple consoles open in one terminal.

Logging In to a Local Console

- Roughly, there are two ways to make yourself known to a Linux server. Sometimes you just sit behind the Linux console and interactively log in from the login prompt that is presented. In other cases, a remote connection is established. The second part of this chapter is about logging in from a remote session; in this part, you learn how to work from a local console.
- If a Linux server boots with a graphical environment (the so-called graphical target), you see a login prompt on which a user name and password can be entered. Many Linux servers do not use a graphical environment at all, though, and are just presenting a text-based console, as shown in Figure 5.1

Logging In to a Local Console



```
CentOS Linux 7 (Core)
Kernel 3.10.0-123.9.3.el7.x86_64 on an x86_64

network login: _
```

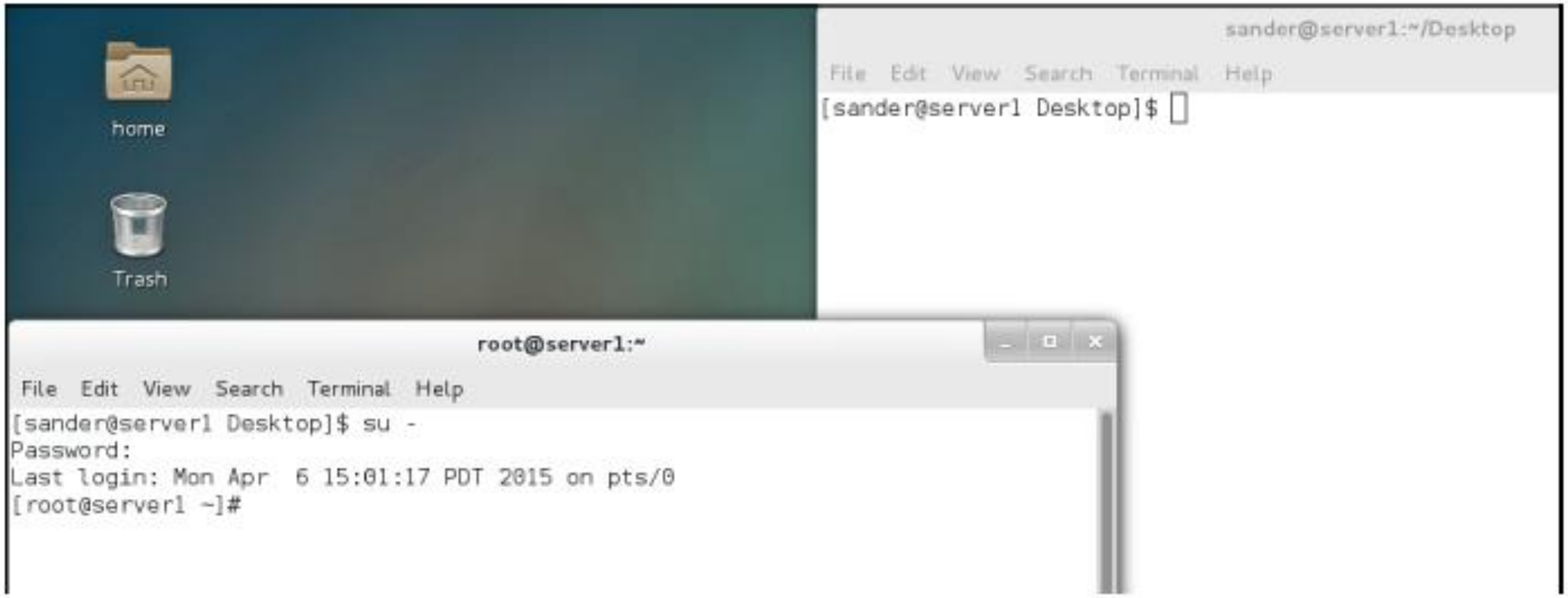
Figure 5.1 Logging in from a text console.

To log in from a text console, you need to know which user account you should use. A user **root** is always available, but using this account to do your work is often not a good idea; the user root has no limitations to access the system and can therefore do a lot of **damage**. A small mistake can have huge impact. Typically, it is a better idea to log in as one of the locally defined users, and there are many reasons to do it.

Switching Between Terminals in a Graphical Environment

- When working in a graphical environment, it is relatively easy to open several different working environments. Just right-click the graphical desktop and select Open in Terminal. This opens several terminal windows as a subshell of the current environment. Because all of these terminals are opened as a subshell, you do not have to log in to each terminal again, and will get access as the same user account that was originally used to log in to the graphical environment (see Figure 5.2).

Switching Between Terminals in a Graphical Environment


home
Trash
sander@server1:~/Desktop
File Edit View Search Terminal Help
root@server1:~
File Edit View Search Terminal Help

```
[sander@server1 Desktop]$
```

```
[sander@server1 Desktop]$ su -
Password:
Last login: Mon Apr  6 15:01:17 PDT 2015 on pts/0
[root@server1 ~]#
```

Figure 5.2 Using different terminal windows from the graphical environment.

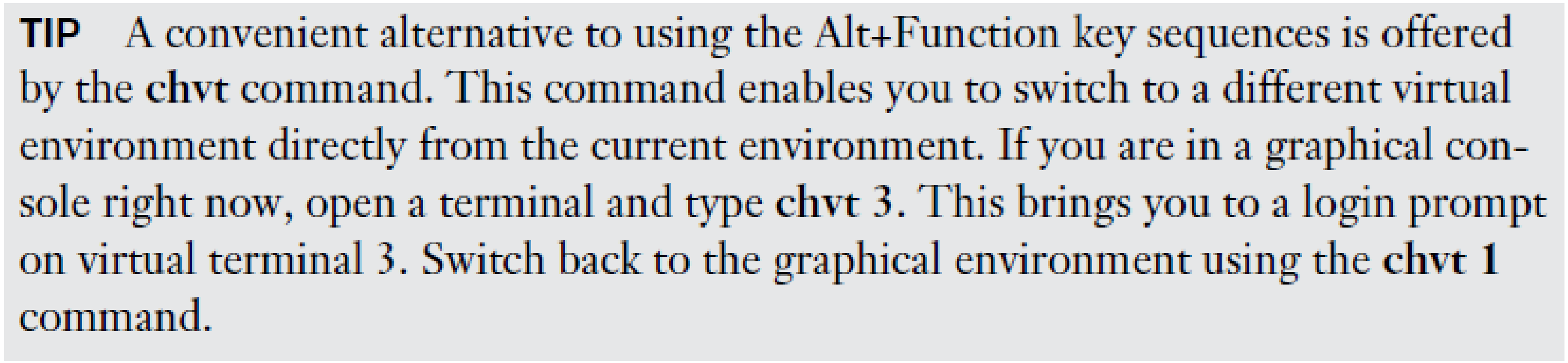
Exercise 5.1 Working from Several Terminal Windows Simultaneously

1. Start your computer and make sure to log in as non-root user account from the graphical log in window that is presented. You should have a local user with the name user and the password password that you can use for this purpose.
2. Right-click an empty spot on the graphical desktop, and from the drop-down menu select Open in Terminal.
3. Repeat step 2 so that you have two terminal windows that are open simultaneously.
4. From one of the terminal windows, type the command **su -** and enter the password of the root user. Then, type **tail -f /var/log/secure**. This opens a trace on the file /var/log/secure, where you can monitor security events in real time.
5. From the other terminal windows, type **su -**. When asked for a password, you normally enter the password for the user root. Enter a wrong password.
6. Now look at the terminal where the trace on /var/log/secure is still open. You will see that an error message has been written to this file.
7. Use the **Ctrl+C** key sequence to close the **tail -f** session on the /var/log/secure file.

Working with Multiple Terminals in a Nongraphical Environment

- In the previous section, you learned how to work with multiple terminals in a graphical environment. This is relatively easy because you just have to open a new terminal window. In a nongraphical environment, you just have one terminal interface that is available and that makes it a little more difficult to work in different user shell environments.
- To offer an option that makes working from several consoles from the same server possible, Linux uses the concept of a virtual terminal. This feature allows you to open six different terminal windows from the same console at the same time. To open these terminal windows, you can use the key sequences **Alt+F1** through **Alt+F6** .

Working with Multiple Terminals in a Nongraphical Environment



TIP A convenient alternative to using the Alt+Function key sequences is offered by the **chvt** command. This command enables you to switch to a different virtual environment directly from the current environment. If you are in a graphical console right now, open a terminal and type **chvt 3**. This brings you to a login prompt on virtual terminal 3. Switch back to the graphical environment using the **chvt 1** command.

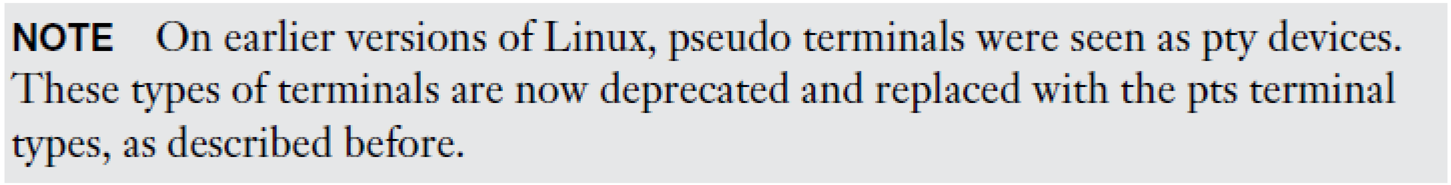
Of these virtual consoles, the first one is used as the default console. It is commonly known as the *virtual console tty1* , and it has a corresponding device file in the /dev directory that has the name /dev/tty1. The other virtual consoles also have corresponding device files, which are numbered /dev/tty1 through /dev/tty6

Working with Multiple Terminals in a Nongraphical Environment

- When working from a graphical environment, it is also possible to open different virtual consoles. Because the combinations between the Alt key and the Function keys typically already have a meaning in the graphical environment, you need to use **Ctrl+Alt+Function key** instead.
- So do not use **Alt+F2** to open /dev/tty2 from a graphical environment, but use **Ctrl+Alt+F2** . To get back to the graphical console, you can use the **Alt+F1** key sequence.
- The **Alt+F6** and the **Ctrl+Alt+F6** key sequences are essentially the same. It just is important to use the Ctrl key as well when going from a GUI to a text environment. To go back from the text environment to the GUI environment, using the Ctrl key is optional.

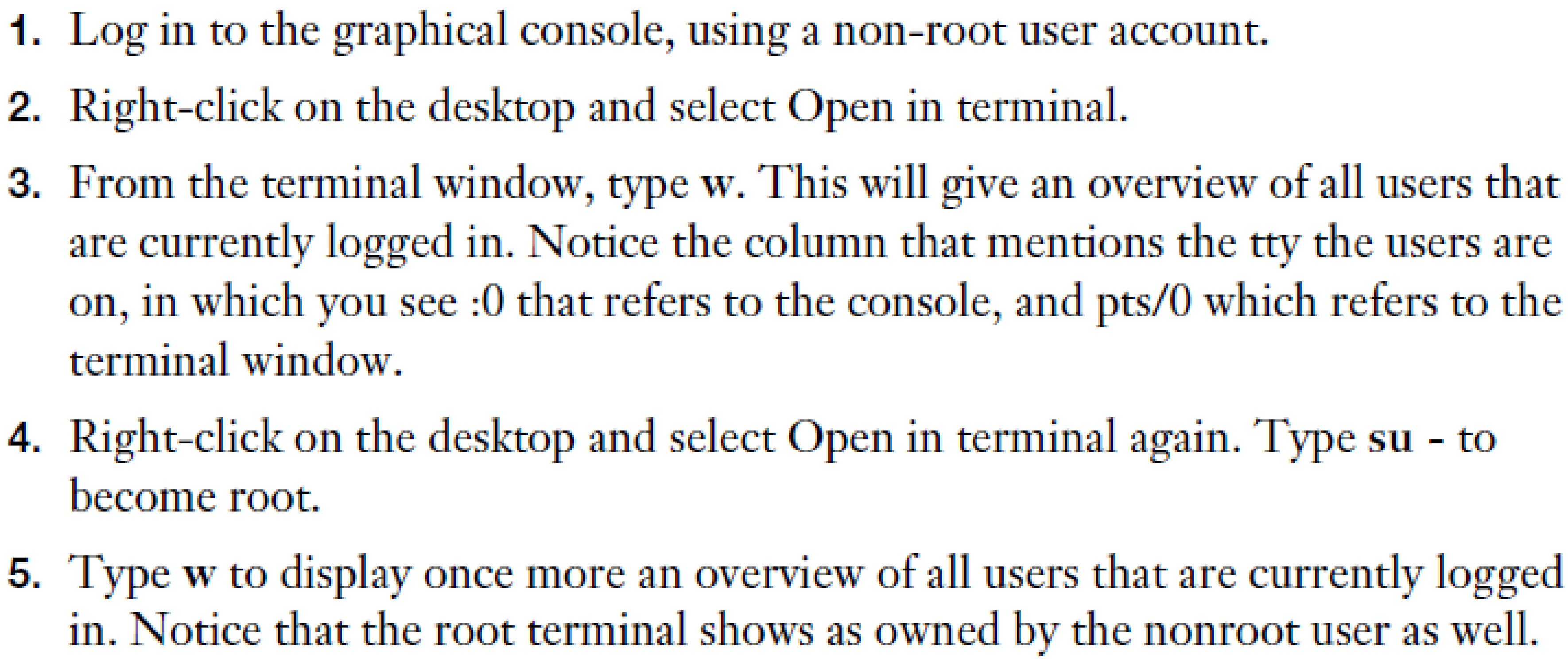
Understanding Pseudo Terminal Devices

- Every terminal used in a Linux environment has a device file associated with it. You’ve just learned that terminals that are started in a nongraphical environment are typically referred to through the devices /dev/tty1 through /dev/tty6.
- For terminal windows that are started from a graphical environment, pseudo terminals are started. These pseudo terminals are referred to using numbers in the /dev/pts directory.
- So, the first terminal window that is started from a graphical environment shows as /dev/pts/1, the second terminal windows is /dev/pts/2, and so on.



NOTE On earlier versions of Linux, pseudo terminals were seen as pty devices. These types of terminals are now deprecated and replaced with the pts terminal types, as described before.

Exercise 5.2 Working with Pseudo Terminals



1. Log in to the graphical console, using a non-root user account.
2. Right-click on the desktop and select Open in terminal.
3. From the terminal window, type **w**. This will give an overview of all users that are currently logged in. Notice the column that mentions the tty the users are on, in which you see :0 that refers to the console, and pts/0 which refers to the terminal window.
4. Right-click on the desktop and select Open in terminal again. Type **su -** to become root.
5. Type **w** to display once more an overview of all users that are currently logged in. Notice that the root terminal shows as owned by the nonroot user as well.

At this point, you know how to work with the console, terminals, virtual terminals, and pseudo terminals. In the section “Using SSH and Related Utilities” later in this chapter, you use SSH to open terminal sessions to your server. These sessions show as pseudo terminals as well.

Using SSH and Related Utilities

- In the previous sections in this chapter, you learned how to access a terminal if you have direct access to the server console. Many administrators work with servers that are not physically accessible. To manage these servers, Secure Shell (SSH) is normally used. In this section, you learn how to work with SSH.
- On modern Linux distributions, Secure Shell is the common method to gain access to other machines over the network. In SSH, cryptography is used to ensure that you can be sure that you are connecting to the intended server. Also, traffic is encrypted while transmitted.

Accessing Remote Systems Using SSH

- To access a server using SSH, two components are needed.
- On the remote server that you want to access, the **sshd** service must be running and offering services at port 22, and it should not be blocked by the firewall. If the SSH port is open, you can access it using the **ssh** command from the command line.
- The **ssh** command by default tries to reach the ssh process on the server port 22. If you have configured the sshd process to offer its services on a different port, use **ssh -p** followed by the port number you want to connect to.

Accessing Remote Systems Using SSH

- When remotely connecting to a Linux server, the SSH client tries to do that as the user account you are currently logged in with on the local machine.
- If you want to connect using a different user account, you can specify the name of this user on the command line, in the user@server format. If, for instance, you want to establish an SSH session as user root to a remote server, type **ssh root@remoteserver** .
- An alternative way of specifying the user account is by using the option **-l username** .
- So, **ssh remoteserver -l root** will also try to establish a root session to the remote server.

Exercise 5.3 Using SSH to Log In to a Remote Server

- In this exercise, you log in to a remote server using SSH. This exercise assumes that a remote server is available and reachable. In this exercise, server1 is used as the local server, and server2 is the remote server on which the SSH process should be up and running.
- If you cannot access a remote server to perform the steps in the exercise, you might alternatively replace server2 with localhost. It is obvious that by doing so you will not log in to a remote server, but you still use the **ssh** command to connect to an sshd process.

Exercise 5.3 Using SSH to Log In to a Remote Server

1. Open a root shell on server2. Type **systemctl status sshd**. This should show you that the sshd process is currently up and running.
2. To avoid any firewall-related problems, type **systemctl stop firewalld**. (In Chapter 22, "Configuring a Firewall," you learn how to configure the firewall to allow for SSH traffic).
3. Type **ip a | grep 'inet '**. (Notice the space between **inet** and the closing quote.) Notice the IPv4 address your server is currently using. In the rest of this exercise, it is assumed that server2 is using IP address 192.168.122.220. Replace that address with the address that you have found here.
4. Open a shell as a nonprivileged user on server1.
5. On server1, type **ssh 192.168.122.220 -l root**. This connects to the sshd process on server2 and opens a root shell.

Exercise 5.3 Using SSH to Log In to a Remote Server

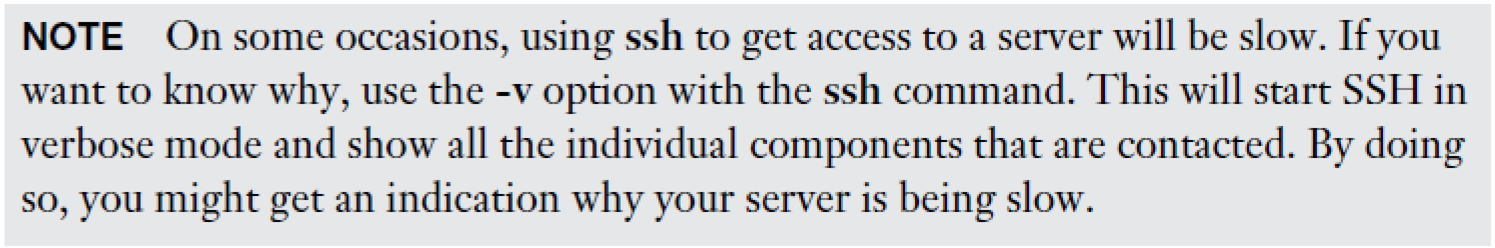
6. Before being prompted for a password, you see a message indicating that the authenticity of host 192.168.122.220 cannot be established (see Listing 5.1). This message is shown because the host you are connecting to is not yet known on your current host, which might involve a security risk. Type **yes** to continue.
7. When prompted, enter the root password. After entering it, you now are logged in to server2.
8. Type **w**. Notice that the SSH session you have just opened shows as just another pseudo terminal session.
9. Type **exit** to close the SSH session.

Listing 5.1 When Logging In to a Remote Server for the First Time, You See a Security Message

```
[root@server1 ~]# ssh 192.168.4.220 -l root
The authenticity of host '192.168.4.220 (<no hostip for proxy
command>)' can't be established.
ECDSA key fingerprint is 35:64:36:f8:ac:4f:8a:94:aa:6e:4b:85:ed:76:0a:
eb.
Are you sure you want to continue connecting (yes/no)?
```

Accessing Remote Systems Using SSH

- You have noticed that while using SSH to connect to the remote server a security message was displayed. This is because the remote server has never been contacted before and therefore there is no way to verify the identity of the remote server. After connecting to the remote server, a public key fingerprint is stored in the file ***~/.ssh/ known_hosts*****.**
- The next time you connect to the same server, this fingerprint is checked with the encryption key that was sent over by the remote server to initialize contact. If the fingerprint matches, you will not see this message anymore.
- In some cases, the remote host key fingerprint does not match the key fingerprint that is stored locally. That is a potentially dangerous situation. Instead of being connected to the intended server, you might be connected to the server of an evildoer. It does, however, also happen if you are connecting to an IP address that you have been connected to before, but this IP address is now in use by a different server. In that case, you just have to remove the key fingerprint from the ***~/.ssh/known_hosts*** file on the client computer.


NOTE On some occasions, using **ssh** to get access to a server will be slow. If you want to know why, use the **-v** option with the **ssh** command. This will start SSH in verbose mode and show all the individual components that are contacted. By doing so, you might get an indication why your server is being slow.

Using Graphical Applications in an SSH Environment

- From an SSH session, by default you cannot start graphical applications. That is because of security; a remote host cannot draw screens on your computer without specific permission to do that. There are two requirements for starting graphical applications through an SSH connection:
  - An X server must be running on the client computer. The X server is the software component that creates the graphical screens.
  - The remote host must be allowed to display screens on the local computer.
- The easiest way to allow the remote host to draw graphical screens on your computer is by adding the **-X** option to the **ssh** command. So, use **ssh -X linda@server2** if you want to connect as linda to server2, and also be able to start graphical applications.

Common *ssh* Options

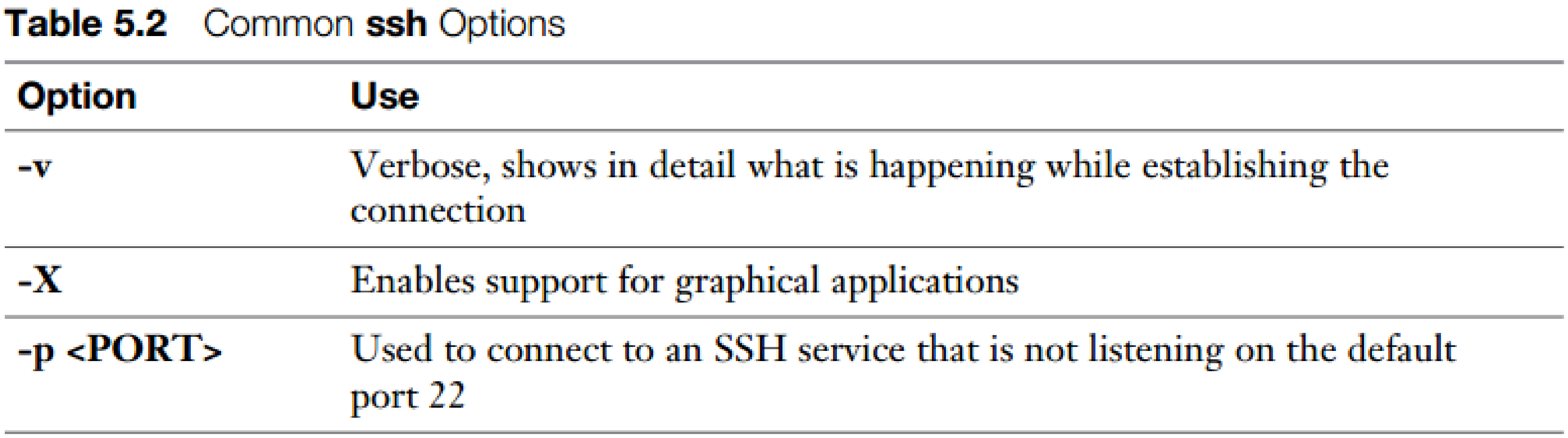


Table 5.2 Common **ssh** Options

Option	Use
-v	Verbose, shows in detail what is happening while establishing the connection
-X	Enables support for graphical applications
-p <PORT>	Used to connect to an SSH service that is not listening on the default port 22

As an administrator, you can also create a systemwide configuration that allows you to use “X forwarding,” which is starting graphical applications through an SSH session. As root, open the configuration file **/etc/ssh/ssh_config** and make sure it includes the following line:

```
ForwardX11 yes
```

The next time you use the **ssh** command, X forwarding will be available by default.

Securely Transferring Files Between Systems

- If a host is running the sshd service, that service can also be used to securely transfer files between systems. To do that, you can use the **scp** command. This command is very similar to the **cp** command, which is used to copy local files, but it does include an option that allows it to work with remote hosts. You can use **scp** to copy files and subdirectories to remote hosts, and subdirectories as well.
- To copy, for instance, the /etc/hosts file to the /tmp directory on server2 using your current user account, use the following command:

```
scp /etc/hosts server2:/tmp
```

Securely Transferring Files Between Systems

- If you want to connect to server2 as user root to copy the /etc/passwd file to your home directory, you use the following command:

  ```
  scp root@server2:/etc/passwd ~
  ```

- You can also use **scp** to copy an entire subdirectory structure. To do so, use the **–r** command, as in the following command:

  ```
  scp -r server2:/etc/ /tmp
  ```

- Notice that the **scp** command can be configured to connect to a non-default SSH port also. It is a bit confusing, but to do this with the **scp** command, you need the **-P** option followed by the port number you want to connect to. Notice that **ssh** uses **-p** (lowercase) to specify the port it needs to connect to; the **scp** command uses an uppercase **-P**.

Configuring Key-Based Authentication for SSH

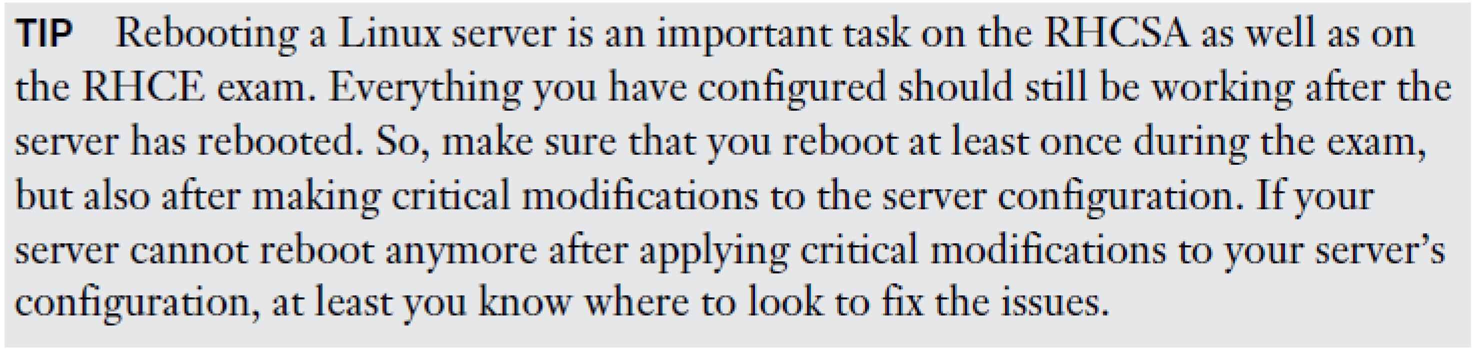
- If SSH is used on the Internet, it might not be a good idea to allow password logins. To make SSH a bit more secure, it will always first try whether login using public/private keys is possible. Only if that is not possible is a password login used. The only thing you need to do to enable key-based login is to create a key pair.
- When using public/private key-based authentication, the user who wants to connect to a server generates a public/private key pair. The private key needs to be kept private and will never be distributed. The public key is stored in the home directory of the target user on the SSH server.
- When authenticating using key pairs, the user generates a hash derived from the private key. This hash is sent to the server, and if on the server it proves to match the public key that is stored on the server, the user is authenticated.
- To create a key pair, use the **ssh-keygen** command. The **ssh-copy-id** command is next used to copy the public key over to the target server.

Exercise 5.4 Connecting to a Remote Server with Public/Private Keys

1. On server1, open a root shell.
2. Type **ssh-keygen**. When asked whether you want to use a passphrase, press **Enter** to use the passphrase-less setup.
3. When asked for the filename in which to store the (private) key, accept the default filename ~/.ssh/id_rsa.
4. When asked to enter a passphrase, press **Enter** twice.
5. The private key will now be written to the ~/.ssh/id_rsa file and the public key is written to the ~/.ssh/id_rsa.pub file.
6. Use **ssh-copy-id server2** to copy the public key you have just created over to server2. You are then asked for the password on the remote server one last time.
7. After copying the public key, verify that it can actually be used for authentication. To do this, type **ssh server2**. You should now authenticate without having to enter the password for the remote user account.

Booting, Rebooting, and Shutting Down Systems

- As an administrator of a Linux server, you occasionally have to reboot the Linux server. Rebooting a server is not often a requirement, but it can make your work a lot easier because it will make sure that all processes and tasks that were running on your server have re-read their configurations and initialized properly.



TIP Rebooting a Linux server is an important task on the RHCSA as well as on the RHCE exam. Everything you have configured should still be working after the server has rebooted. So, make sure that you reboot at least once during the exam, but also after making critical modifications to the server configuration. If your server cannot reboot anymore after applying critical modifications to your server's configuration, at least you know where to look to fix the issues.

Booting, Rebooting, and Shutting Down Systems

- For an administrator who really knows Linux very thoroughly, rebooting a server is seldom necessary. Experienced administrators can often trigger the right parameter to force a process to reread its configurations. There are some cases, though, that even experienced Linux administrators have to reboot:
  - To recover from serious problems such as server hangs and kernel panics
  - To apply kernel updates
  - To apply changes to kernel modules that are used and because of that cannot be reloaded easily

Booting, Rebooting, and Shutting Down Systems

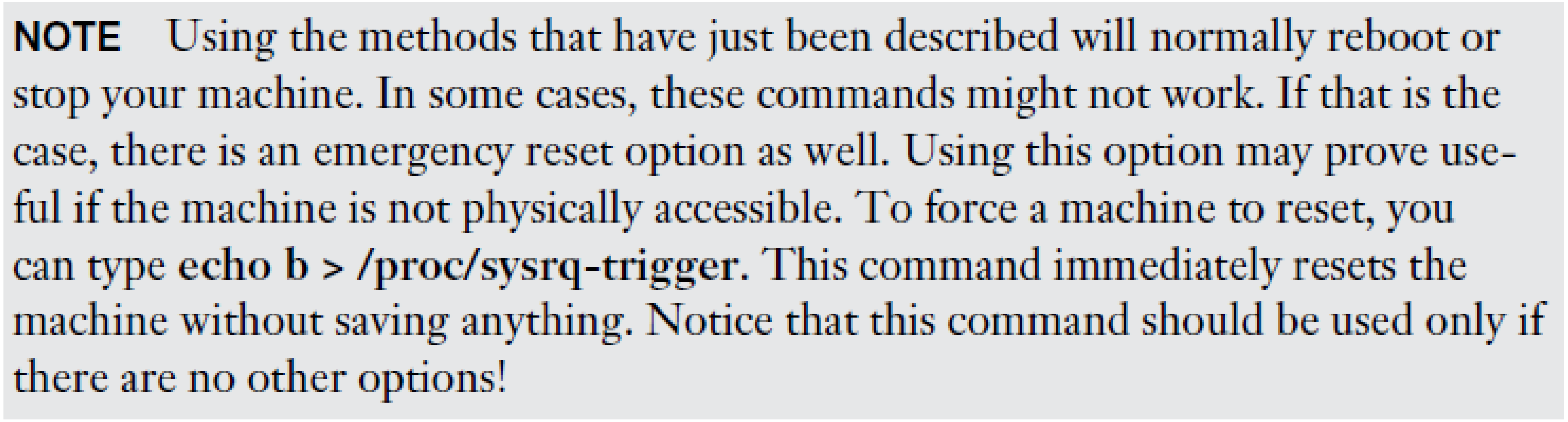
- When a server is rebooted, all processes that are running need to shut down properly. If the server is just stopped by pulling the power plug, much data will typically be lost. That is because processes that have written data do not typically write that data directly to disk, but instead store it in memory buffers from where it is committed to disk when it is convenient for the operating system.
- To issue a proper reboot, the systemd process has to be alerted. The systemd process is the first process that was started when the server was started, and it is responsible for managing all other processes, directly or indirectly. As a result, on system reboots or halts, the systemd process needs to make sure that all these processes are stopped. To tell the systemd process this has to happen, a few commands can be used:

Booting, Rebooting, and Shutting Down Systems

- **systemctl reboot** or **reboot**
- **systemctl halt** or **halt**
- **systemctl poweroff** or **poweroff**

- When stopping a machine, you can use the **systemctl halt** or the **systemctl poweroff** commands. The difference between these two commands is that the **systemctl poweroff** command talks to power management on the machine to shut off power on the machine. This often does not happen when using **systemctl halt** .

Booting, Rebooting, and Shutting Down Systems



NOTE Using the methods that have just been described will normally reboot or stop your machine. In some cases, these commands might not work. If that is the case, there is an emergency reset option as well. Using this option may prove useful if the machine is not physically accessible. To force a machine to reset, you can type **echo b > /proc/sysrq-trigger**. This command immediately resets the machine without saving anything. Notice that this command should be used only if there are no other options!

Summary

- In this chapter, you learned how to connect to Red Hat Enterprise Linux 7. You learned the difference between consoles, terminals, and shells, and you learned how to set up terminal sessions locally as well as remotely. You also learned how to use SSH to connect to a remote server, and how to securely copy files between servers.

Define Key Terms

- Define the following key terms:
  - Console
  - Terminal
  - Tty
  - Login shell
  - Subshell
  - Reboot
  - Systemd
  - Key-based login
  - Public key
  - Private key

Lab 5.1

1. Log in to the local console on server1. Make sure that server1 does *not* show a graphical interface anymore, but just a text-based login prompt. Log in from that environment and activate tty6. From tty6, switch back on the graphical interface and use the correct key sequence to go to the graphical interface.

2. Set up SSH based authentication. From server2, use SSH to connect to server1. Make sure that graphical applications are supported through the SSH session. Also set up key-based authentication so that no password has to be entered while connecting to the remote server.

Q&A